НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМ. ІГОРЯ СІКОРСЬКОГО»

Факультет Інформатики та обчислювальної техніки
(повне найменування інституту, факультету)

Обчислювальної техніки
(повна назва кафедри)

До захисту допущено
Завідувач кафедри
_Сергій Стіренко___
(підпис) (ініціали, прізвище)
"___"_____2020р.

Дипломний проект

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення комп'ютерних систем»

спеціальності **121 «Програмна інженерія»**
(код і назва)

на тему: «Система обробки та візуалізації електрофізіологічних даних»

Виконав (-ла): студент (-ка) _IV_ курсу, групи _ІП-62__
(шифр групи)

_Сабецька Марина Віталіївна_____
(прізвище, ім'я, по батькові) (підпис)

Керівник _проф. д.т.н. Сімоненко В.П._____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант нормоконтроль проф. д.т.н. Сімоненко В.П._____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМ. ІГОРЯ СІКОРСЬКОГО”

Факультет інформатики та обчислювальної техніки
(повне найменування інституту, факультету

Обчислювальної техніки

Рівень вищої освіти **перший (бакалаврський)**

Спеціальність **121 «Програмна інженерія»**

Освітньо-професійна програма **«Інженерія програмного забезпечення комп’ютерних систем»**

ЗАТВЕРДЖУЮ

Завідувач кафедри

_Сергій Стіренко_____
(прізвище ініціали) (підпис)

“___”_____2020 р.

ЗАВДАННЯ

на бакалаврський дипломний проект студенту

__Сабецькій Марині Віталіївні_____
(прізвище, ім’я, по батькові)

1. Тема проекту «*Система обробки та візуалізації електрофізіологічних даних»* керівник проекту *Сімоненко Валерій Павлович, доктор технічних наук, професор* _____,
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від “07” 05 2019 року №1180-С
1. Термін здачі студентом закінченого проекту _____
3. Вихідні дані до проекту *нормативна документація, файли експериментальних даних, наукова література, бібліотеки підпрограм.*
4. Зміст пояснювальної записки (перелік питань, які розробляються):
опис предметної області та напрямків досліджень, огляд існуючих розрахункових комплексів, аналіз параметрів і методів обробки електрофізіологічних даних, розробка системи обробки і візуалізації.

5. Перелік графічного матеріалу: схема роботи системи, схема взаємодії модулів, схема алгоритму апроксимації.

6. Консультант проекту , з вказівкою розділів роботи, які до них вносяться

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Нормоконтроль	Сімоненко В.П.		

7. Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Найменування етапів дипломного проекту	Строк виконання етапів проекту	Примітки
1.	*Затвердження теми роботи*	*3.02.2020*	
2.	*Аналіз завдання*	*3.02.2020 - 13.04.2020*	
3.	*Ознайомлення з напрямком досліджень*	*13.04.2020 - 19.04.2020*	
4.	*Аналіз існуючих систем*	*13.04.2020 - 3.05.2020*	
5.	*Розробка системи*	*13.04.2020 - 17.05.2020*	
6.	*Написання пояснювальної записки*	*20.04.2020 - 17.05.2020*	
7.	*Оформлення ДПБ*	*20.04.2020 - 26.05.2020*	
8.	*Нормоконтроль*	*26.05.2020*	
9.	*Попередній захист*	*26.05.2020*	
10.	*Захист*		

Студент-дипломник _____
(підпис)

Керівник роботи _____
(підпис)

АНОТАЦІЯ

У бакалаврському проекті виконано роботу з розробки системи обробки даних електрофізіологічних досліджень нервових клітин методом patch-clamp. Враховуючи конфігурацію експерименту, визначено основні параметри і способи їх розрахунку. З бібліотек для мови програмування Python обрано відповідні функції для розрахунку, а також апроксимації та фільтрації даних.

Для зручності взаємодії користувача з системою за допомогою фреймворку Qt створено графічний інтерфейс, в якому користувач може вказати всі необхідні для обробки параметри.

Використовуючи дану програму можна провести обробку даних, створити зображення для візуалізації отриманих результатів, вивести на екран для попереднього перегляду і зберегти в растрових і векторних форматах.

Дипломний проект включає опис напрямку досліджень, розрахункові формули, відповідні алгоритми реалізовані мовою Python та графічне представлення роботи системи.

ABSTRACT

In this Bachelor's degree project the work on the development of a data processing system for electrophysiological studies of nerve cells using the patch-clamp method has been performed. Given the configuration of the experiment, the main parameters and methods of their calculation have been determined. Using the libraries of the Python programming language, the appropriate functions for calculation, as well as approximation and filtering of data, were selected.

For the better user interaction with the system, the graphic interface, in which the user can specify all the parameters necessary for the processing, was created by utilizing Qt framework.

By using this program, the user will be able to process data, create images to visualize the results, display the results for preview and save the images in both raster and vector formats.

This diploma project includes a description of the research, calculation formulas, appropriate algorithms implemented in Python and a graphical representation of the system.

Формат	Зона	Поз.	Позначення	Найменування	Кіл.	Прим.
				Документація		
А4		1	ІАЛЦ 467800.002 ТЗ	Технічне завдання	4	
А4		2	ІАЛЦ 467800.003 ПЗ	Пояснювальна записка	79	
А3		3	ІАЛЦ 467800.004 Д1	Схема роботи системи	1	
А4		4	ІАЛЦ 467800.004 Д2	Схема взаємодії модулів	1	
А4		5	ІАЛЦ 467800.004 Д3	Структурна схема програми	1	
А4		6	ІАЛЦ.467800.005 Д4	Лістинг програми	10	
А4		7	ІАЛЦ.467800.006 Д5	Опис програми	6	

Зм.	Арк.	№ докум.	Підп.	Дата.	ІАЛЦ 467800.001 ВП		
Розробив		Сабецька			Система обробки та візуалізації електрофізіологічних даних	Літ.	Арк. 1 / Аркушів
Перевірив		Сімоненко					
Н.контр.		Сімоненко			Відомість дипломного проекту	НТУУ «КПІ ім. Ігоря Сікорського», ФІОТ, ІП-62	
Затв.		Стіренко					

ТЕХНІЧНЕ ЗАВДАННЯ

до дипломного проекту
освітньо-кваліфікаційного рівня бакалавр

на тему: “Система обробки та візуалізації електрофізіологічних даних”

Київ – 2020 року

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Дане технічне завдання поширюється на розробку Системи обробки та візуалізації електрофізіологічних даних.

Область застосування: електрофізіологія; використання науковцями які працюють методикою patch-clamp з обладнанням від Molecular Devices.

2. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання роботи кваліфікаційно-освітнього рівня «бакалавр комп'ютерної інженерії», затверджене кафедрою обчислювальної техніки Національного технічного Університету України «Київський Політехнічний інститут ім. Ігоря Сікорського».

3. МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою проекту є розробка системи обробки та візуалізації електрофізіологічних даних, для оптимізації дослідницької роботи.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом розробки є науково-технічна література з теорії і практики програмування та з електрофізіології, керівництво з експлуатації електрофізіологічного обладнання, нормативні документи та публікації в Інтернеті з даних питань.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до продукту:

- пакетна обробка даних;
- перехоплення помилок що виникають внаслідок введення неправильних параметрів чи невідповідних вхідних даних;
- кросплатформність та незалежність від ОС;
- можливість попереднього перегляду та редагування графіків;
- наявність набору налаштувань за замовчуванням.

5.2. Вимоги до програмного забезпечення:

- ОС MS Windows від XP до 10, Linux;
- Python від версії 3.0;
- бібліотеки NumPy, Matplotlib, pyABF, PyQt5.

5.3. Вимоги до апаратної частини

- оперативна пам’ять від 256 Мбайт;
- вільне місце на жорсткому диску від 100 Мбайт;
- роздільна здатність монітора від 1024×768 пікселів;
- платформа x86, ARM та ін., що підтримують Python і відповідні бібліотеки.

6. ЕТАПИ РОЗРОБКИ

1) Аналіз завдання і основних вимог	3.02 - 13.04.2020
2) Проектування системи	6.04 – 13.04.2020
3) Програмування основних модулів системи	13.04 – 3.05.2020
4) Компонування	3.05 – 10.05.2020
5) Виправлення помилок і налагодження	10.05 – 17.05.2020
6) Оформлення документації	17.05 - 26.05.2020

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проекту
освітньо-кваліфікаційного рівня бакалавр

на тему: “Система обробки та візуалізації електрофізіологічних даних”

Київ – 2020 року

ЗМІСТ

СПИСОК СКОРОЧЕНЬ

ОС – операційна система.

ПЗ – програмне забезпечення.

ABF - Axon Binary File – бінарний файл Аксону, формат файла підсилювача.

CAP - Cell Attached Pipette – піпетка прикладена до клітини, конфігурація експерименту.

DAQ - Data Aquisition System – система збору даних.

EPS - Encapsulated PostScript - графічний формат, який описано мовою PostScript.

GIF - Graphics Interchange Format – «формат обміну зображеннями», растровий формат.

GUI - Graphical User Interface – графічний інтерфейс користувача.

HTML - HyperText Markup Language – мова розмітки гіпертексту.

IDE - Integrated Development Environment – інтегроване середовище розробки.

PGML - Precision Graphics Markup Language – мова точної розмітки графіки.

PNG - Portable Network Graphics – растровий графічний формат.

RGB - Red, Green, Blue – адаптивна колірна модель.

SVG - Scalable Vector Graphics - масштабована векторна графіка, формат файлів.

VML - Vector Markup Language – мова розмітки векторів, формат експорту даних.

WC – Whole Cell – конфігурація експерименту з цілою клітиною.

XHTML - Extensible Hypertext Markup Language – розширювана мова розмітки гіпертексту.

XML - eXtensible Markup Language – розширювана мова розмітки, стандарт побудови мов розмітки.

ВСТУП

Для досліджень в області нейрофізіології дослідницькими лабораторіями в Україні та закордоном використовується пакет збору даних pCLAMP. Він зберігає дані в бінарні файли, які можна обробляти спеціалізованими обробними комплексами, або створеними самостійно із застосуванням відкритих бібліотек. Для більшості загальних задач підходить перший варіант. У випадку більш специфічних варіантів обробки даних без створення власних програм роботу доводиться виконувати вручну, що займає досить багато часу.

Для подальшого використання та ведення статистики даних досліджень обробки спинним мозком больової інформації ці дані необхідно обробити по заданому шаблону. Всі задіяні в обробці даних мають робити це за уніфікованим сценарієм, чого важко досягти, коли кожен учасник використовує різні програмні продукти. Крім того, кожна з програм має свою мову макросів, тому для створення макросів для пакетної обробки електрофізіологам необхідно вивчити відповідну мову, або вже мати додаткові навички.

Розроблювана програма повинна виконувати зчитування та пакетну обробку бінарних файлів, зберігаючи отримані після обробки графіки у векторні та растрові формати. Для кожного з файлів потрібно створювати окремий графік з заданими параметрами. Визначення якості виконаного експерименту і характеристик клітини відбувається завдяки так званому «Тесту мембрани», який програма повинна створити, де відображені необхідні для цього параметри клітини, за якими науковець робить відповідні висновки.

Підсилювач зчитуваного струму та напруги може мати два канали запису для двох клітин, нервів, або клітини і нерву, як в нашому варіанті. В

цьому випадку програма повинна виводити дані з двох каналів на один графік з однаковими параметрами і масштабом. Для порівняння часу появи події на різних треках (записах) одного файлу графіки струму/напруги потрібно відобразити з вертикальним зміщенням відносно один одного. При появі шуму в реєстраціях, його необхідно прибрати використовуючи цифрові фільтри. В межах одного файлу треба розділити записи з різною полярністю стимуляції.

Метою проекту є реалізація всіх зазначених вище задач з можливістю оброблювати дані пакетно і будь-ким. Це значно спростить і пришвидшить дослідну роботу без затрат грошей на платне програмне забезпечення та часу на обробку цих даних науковцями вручну. Це особливо актуально наразі, у зв'язку зі скороченням фінансування науки в Україні та штату наукових працівників Відділу сенсорної сигналізації Інституту фізіології ім. О.О. Богомольця НАН України, де виконуються дані дослідження.

В електрофізіології напрацьовані експериментальні дані необхідно наглядно представляти в наукових статтях та на міжнародних і вітчизняних конференціях. Українські дослідні інститути співпрацюють з інститутами в різних країнах світу, для яких також необхідно демонструвати напрацьовані дані. Це потребує якісного графічного відображення. Стандартний набір функцій більшості використовуваних у нас програм для обробки електрофізіологічних даних робить зображення растрового формату. При масштабуванні такого зображення губиться роздільна здатність, з ним неможливо виконувати геометричні перетворення та багато іншого. Для більшості потреб зображення мають бути векторні.

Під час обробки даних вручну за день може бути виконана обробка близько десятка файлів даних. Якщо автоматизувати цей процес, включаючи пакетну обробку, їх кількість зросте майже в десять разів. Це, відповідно, збільшить ефективність роботи з обробки і залишить більше часу для проведення експериментів і напрацювання даних. Отже, розробка даної системи є актуальною для зазначеної області і економічно доцільною.

РОЗДІЛ 1. ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ ТА НАПРЯМКІВ ДОСЛІДЖЕНЬ

Техніка patch clamp (або метод фіксації потенціалу) - це лабораторний метод в електрофізіології, що використовується для вивчення іонних струмів в окремих ізольованих живих клітинах, тканинних зрізів, або ділянки клітинної мембрани. Методика особливо корисна при дослідженні збудливих клітин, таких як нейрони, кардіоміоцити, м'язові волокна та інші.

Patch clamp можна виконати за допомогою техніки фіксації напруги. У цьому випадку експериментатор контролює напругу на клітинній мембрані і записуються отримані струми. В якості альтернативи можна застосовувати техніку фіксації струму. У цьому випадку струм, що проходить через мембрану, контролюється експериментатором і реєструються в результаті зміни напруги, як правило, у формі потенціалів дії .

Це дозволяє вивчити фундаментальні клітинні процеси, такі як потенціал дії та нервова активність [42].

1.1. Представлення клітини через електричний контур

Електричні властивості клітина має здебільшого завдяки електричним властивостям своєї мембрани. Між внутрішньою та зовнішною стороною мембрани існує різниця потенціалів ΔV - це різниця внутрішньої сторони клітини і зовнішнього середовища, який прийнято за нуль. У нейрона ΔV близько -60 мВ.

Різниця потенціалів у клітинній мембрані генерується білками, які використовують хімічну енергію для переміщення іонів через клітинну мембрани. Заряд розділяється і створюється трансмембранний потенціал. Оскільки ліпідна мембрана є хорошим ізолятором, трансмембранний

потенціал зберігається за відсутності відкритих пор або каналів.

Електрофізіологічне обладнання дозволяє вимірювати різницю потенціалів в біологічних системах. Але також обладнання може вимірювати струм, який є потоком електричного заряду, що проходить через будь-яку точку за одиницю часу.

Два правила про струми зберігаються і в електрофізіологічних явищах:

1) струм зберігається у точці відгалуження;
2) струм завжди тече в повному контурі (рис. 1.1).

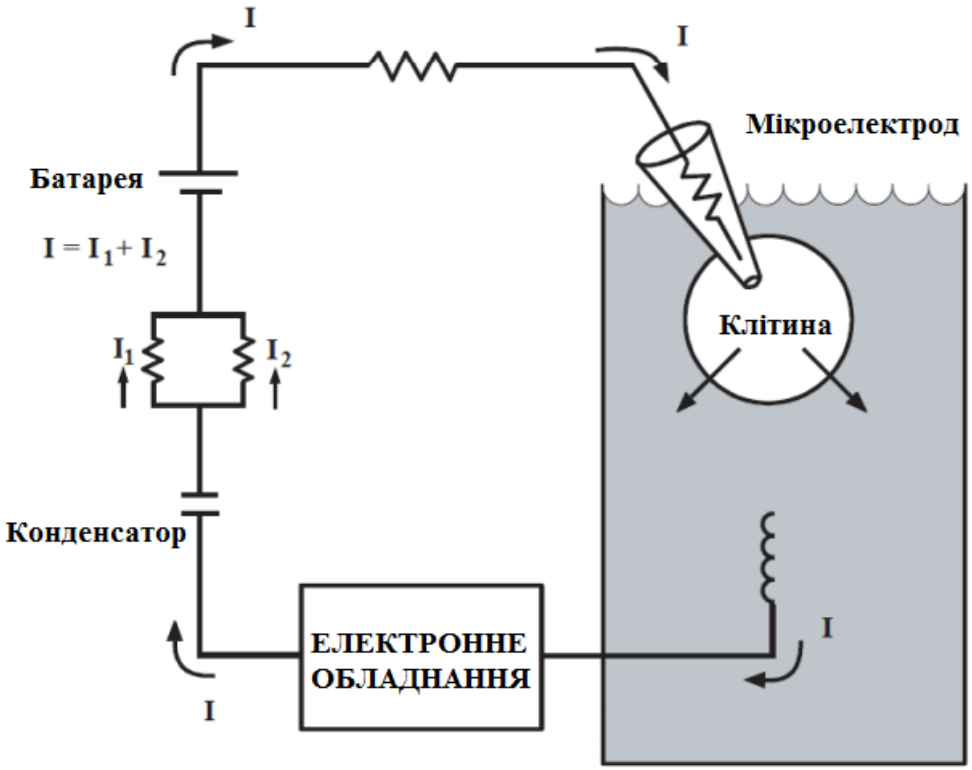


Рисунок 1.1. Типовий електричний ланцюг в електрофізіологічному досліді [14].

Коли декілька іонних каналів у мембрані відкриваються одночасно, загальна провідність є сумою провідності окремих відкритих каналів. Більш точне зображення іонного каналу - це провідник послідовно з двома додатковими елементами ланцюга:

1) перемикач, який представляє затвор каналу,

2) акумулятор, який представляє потенціал зворотного іонного струму

для цього каналу.

Потенціал мембрани в спокої E описує стан без потоку електричного струму через мембрану. Оскільки типова клітинна мембрана в спокої має значно більшу проникність калію ніж для натрію, кальцію чи хлору, потенціал мембрани спокою дуже близький до E_{K+}, рівноважного потенціалу іонів калію [14], що визначають за рівнянням Нернста:

$$E_{K^+} = \frac{RT}{zF} \ln \frac{[K^+]_{зов}}{[K^+]_{вн}}, \quad (1.1)$$

де: E_{K+} - рівноважний потенціал іонів K^+ в вольтах;

R - універсальна газова стала (8,3144 Дж/моль×К);

T - абсолютна температура в кельвінах (К);

z - число елементарних зарядів іонів, що беруть участь в реакції;

F - стала Фарадея (96485 Кл/моль);

$[K^+]_{зов}$ - позаклітинна концентрація іонів калію в ммоль×л;

$[K^+]_{вн}$ - внутрішньоклітинна концентрація іонів калію в ммоль×л [21].

Різниця потенціалів між двома точками пов'язана з потоком струму, провідністю G та струмом I:

$$\Delta V = IR = I/G. \quad (1.2)$$

В експерименті фіксації напруги (voltage-clamp), коли N каналів з коефіцієнтом провідності γ відкриті, загальна провідність - Nγ. Електрохімічна рушійна сила ΔV (різниця мембранного потенціалу і зворотного потенціалу) виробляє струм NγΔV. Коли канали відкриваються та закриваються, їх кількість N змінюється, а тому змінюються струм I. Отже, струм пропорційний кількості відкритих каналів у будь-який момент часу. Кожен канал може розглядатися як приріст провідності γ [14].

1.2. Прилади та обладнання в електричному колі

Електрофізіологічні вимірювання повинні відповідати двом вимогам щодо вимірюваного параметра:

1) необхідний параметр повинен бути точно виміряним,

2) він не повинен бути спотворений.

Перед вимірюванням клітина має потенціал спокою $E_{гр}$, який слід вимірювати внутрішньоклітинним електродом опору R_e. Щоб зрозуміти вплив вимірювального контуру на вимірюваний параметр, приймемо прилад за «ідеальний» вольтметр (з нескінченним опором) з паралельно підключеним кінцевим опором R_{in}, який представляє опір реального вольтметра або вимірювальний контур. Комбінація R_e і R_{in} утворює дільник напруги, так що на вході «ідеального» вольтметра з'являється лише $E_{гр}$; ця частка дорівнює $E_{гр}R_{in}/(R_{in}+R_e)$. Чим більше R_{in}, тим ближче V до $E_{гр}$. Це може бути проблемою, оскільки опір електродів R_e збільшується. Рішення в таком випадку - зробити R_{in} якомога більшим. З іншого боку, найкращий спосіб вимірювання струму - це вставити амперметр. Якщо амперметр має нульовий опір, він не порушить ланцюг, оскільки через нього не буде падіння IR.

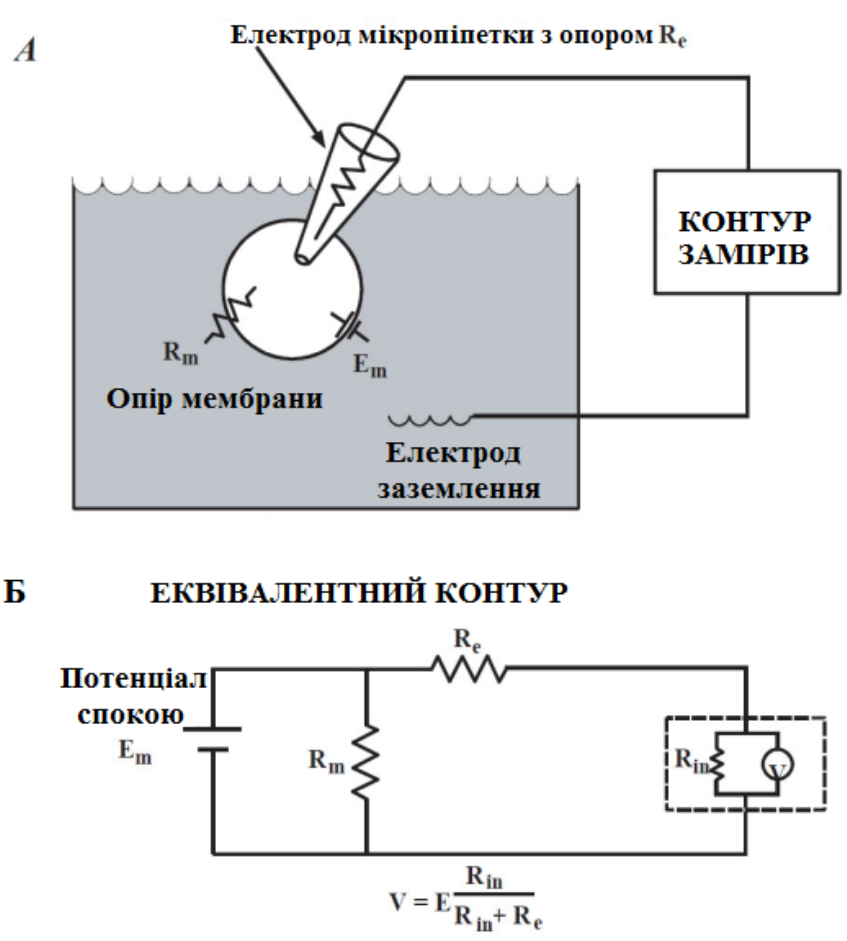


Рисунок 1.2. Вимірювальний контур (А), і його еквівалентна схема (Б)[14].

Електричне поле є властивістю кожної точки в просторі і визначається як пропорційна силі, яку відчуває заряд, поміщений у цю точку. Чим більша різниця потенціалів між двома точками, тим більше поле в кожній точці між ними.

Біологічні мембрани зазвичай мають товщину менше 10 нм. Отже, трансмембранний потенціал спокою близько 100 мВ створює в мембрані електричне поле, що становить близько 10^5 В/см. Типове електрофізіологічне обладнання не може безпосередньо вимірювати ці поля. Однак зміни відчуваються в іонних каналах, чутливих до напруги, і тому електричні поля лежать в основі електричної збудливості мембран.

Ще одним наслідком досить маленької товщини мембрани є те, що вона є відмінним конденсатором. Двошарова ліпідна мембрана є його відмінним наближенням. Ємність C - здатність зберігати заряд Q, коли виникає різниця напруг ΔV:

$$Q = C\Delta V \, . \qquad (1.3)$$

Ємність пропорційна площі та обернено пропорційна відстані, що розділяє дві провідникових пластини. Коли паралельно підключено кілька конденсаторів, це еквівалентно одному великому конденсатору; тобто загальна ємність - це сума їхніх ємностей. Таким чином, ємність мембрани збільшується з розміром клітини. Мембранна ємність зазвичай виражається як значення на одиницю площі; майже всі ліпідні двошарові мембрани клітин мають ємність 1 мкФ/см^2 (0,01 пФ/мкм^2) [14].

1.3. Електрофізіологічні параметри

Заряд зберігається в конденсаторі лише тоді, коли в конденсаторі є зміна напруги. Тому струм, що протікає через ємність C, пропорційний зміні напруги з часом:

$$I = C\frac{\Delta V}{\Delta t} \, . \qquad (1.4)$$

Поки напруга на мембрані залишається постійною, можна ігнорувати

вплив мембранної ємності на струми, що протікають по мембрані через іонні канали. Коли напруга змінюється, крім стаціонарних струмів (steady-state currents) через струмопровідні канали проходять також швидкоплинні ємнісні струми (transient capacitive currents). Ці струми є одим з двох основних залежних від часу електричних властивостей клітини.

Накопичений заряд на мембрані супроводжує потенціал спокою і будь-яка зміна напруги на мембрані супроводжується зміною цього заряду. Якщо на мембрану подається струм каналами або від електрода, цей струм спочатку заряджає мембранну ємність (membrane capacitance), а потім він змінює напругу на мембрані. Формально мембрану можна представити як резистор R паралельно з ємністю C.

Тепер, якщо ми застосуємо імпульс струму до контуру, струм спочатку зарядить ємність (конденсатор), а потім змінить напругу (рис. 1.3):

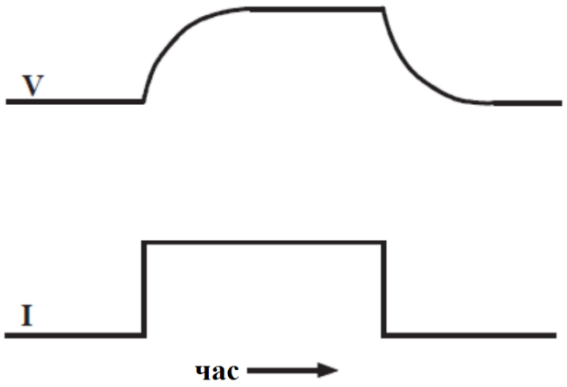


Рисунок 1.3. Реакція паралельного ланцюга RC на крок струму [14].

Напруга V(t) з часом наближається до стаціонарного стану по експоненті:

$$V(t) = V_{inf}(1 - e^{-t/\tau}). \quad (1.5)$$

Стаціонарне значення (steady-state) V_{inf} не залежить від ємності, а визначається поточним значенням I і мембранним опором R:

$$V_{inf} = IR. \quad (1.6)$$

Це закон Ома. Коли мембранна ємність знаходиться у контурі, напруга

не досягається негайно. Натомість, до нього наближається константа часу τ, задана:

$$\tau = RC. \qquad (1.7)$$

Таким чином, константа часу зарядки збільшується, коли збільшується або ємність мембрани, або опір.

В експерименті з фіксацією струму (current-clamp) застосовується відомий постійний або змінний струм і вимірюється зміна мембранного потенціалу, викликана прикладеним струмом. Цей тип експерименту імітує струм, що виробляється синаптичним входом.

В експерименті фіксації напруги (voltage clamp) контролюється мембранна напруга і вимірюється трансмембранний струм, необхідний для підтримки цієї напруги (рис.1.4).

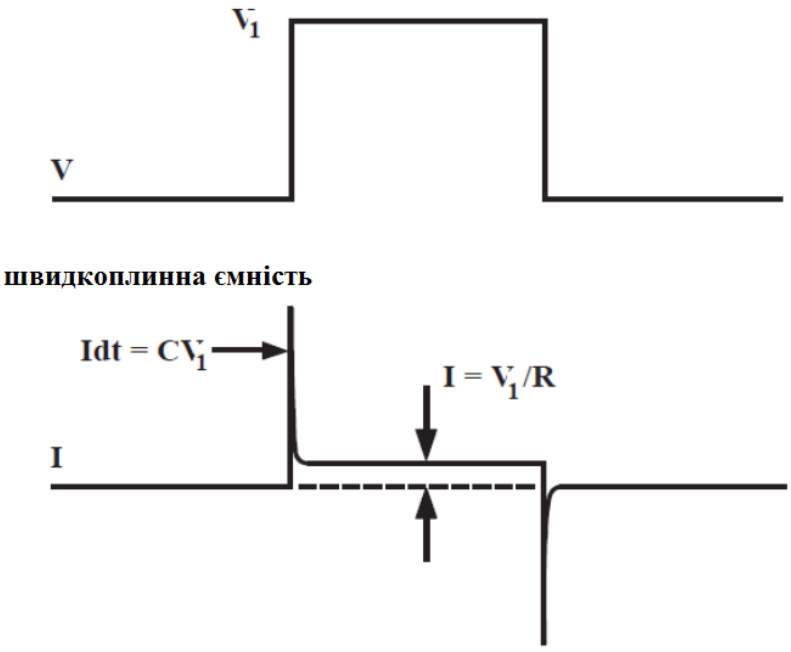


Рисунок 1.4. Типовий експеримент із фіксацією напруги [14].

Patch-clamp - це такий варіант фіксації напруги, що дозволяє визначити струми, які протікають по одноіонних каналах. Він також спрощує вимірювання струмів, що протікають через цілоклітинні (whole-cell) мембрани, особливо для невеликих клітин, в які важко проникнути

електродом.

Patch clamp спирається на два аспекти:

1) виміряні струми дуже малі, порядком пікоамперів в одноканальному записі і до кількох наноампер в цілоклітинному (whole-cell) записі. Через малі струми, особливо в одноканальному записі, поляризація та нелінійність коливань [32] електродів незначна і електрод Ag/AgCl може точно записувати напругу навіть під час проходження струму;

2) електронний амперметр повинен бути таким, щоб уникнути додавання відчутного шуму до струму, який вимірюється.

Успішні електрофізіологічні вимірювання залежать від конструкції електронних приладів, властивостей та процесу виготовлення скляних мікропіпеток. Для успішних записів потрібна щільний контакт між піпеткою та мембраною.

Друга вимога до електрофізіологічних вимірювань (див. 1.2. Прилади та обладнання в електричному колі) говорить, що якість вимірювання залежить від мінімізації спотворень (perturbation). Зобразимо варіант для запису напруги (рис. 1.5) схемою з дільником напруги (рис. 1.6).

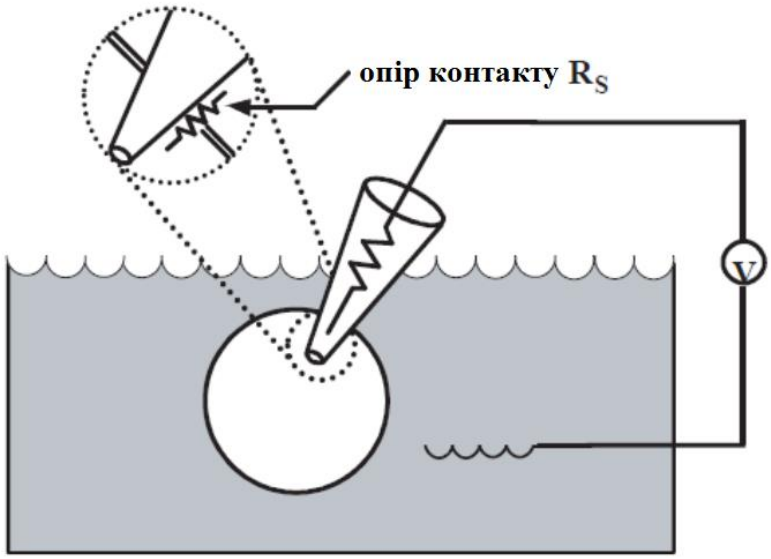


Рисунок 1.5. Внутрішньоклітинне вимірювання електродом. Цей внутрішньоклітинний електрод вимірює потенціал спокою клітини,

мембрана якої містить лише відкриті K^+ канали [14].

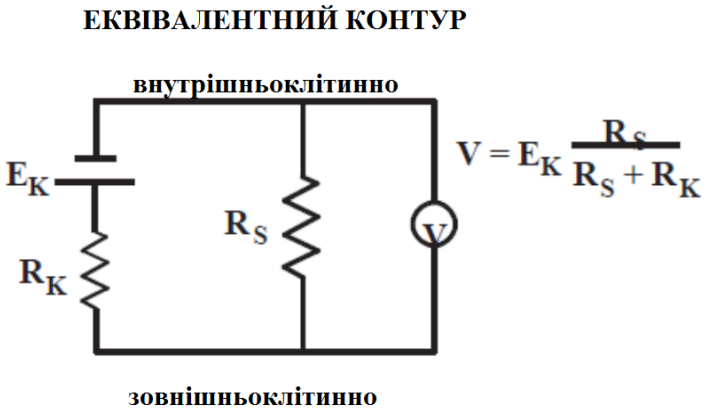


Рисунок 1.6. Зі збільшенням опору контакту R_s опір наближається до значення E_K [14].

У випадку patch-запису струми, що йдуть через контакт з клітиною, не спотворюють вимірювану напругу чи струм, але вони додають до струму шум. Цей шум можна розглядати як шум Джонсона (тепловий шум, який збільшується з провідністю), або з точки зору простої статистики: якщо струм N іонів/мс проходить через відкритий канал, то струм буде коливатися від однієї мілісекунди до другої зі стандартним відхиленням √N. Ці флуктуації створюють шум на одноканальних записах (traces). Якщо додатковий струм протікає паралельно у місці контакту (рис. 1.7), це викликає збільшення стандартних відхилень [14].

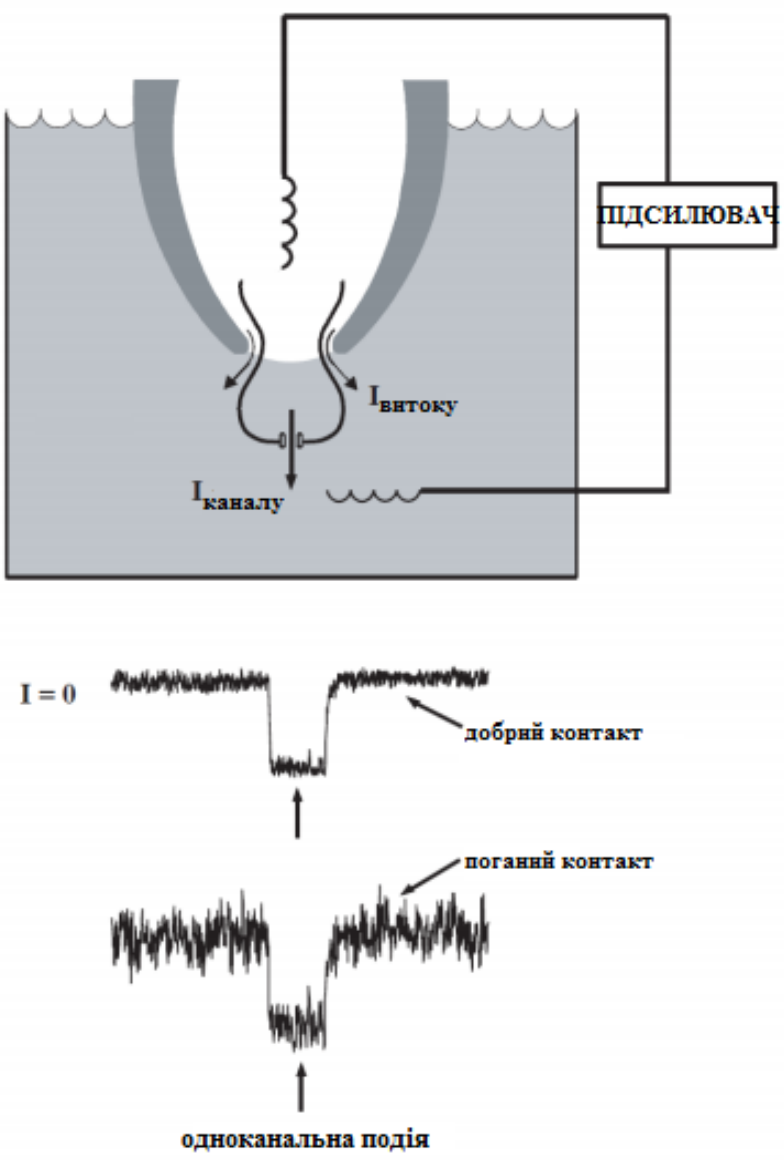


Рисунок 1.7. Хороші та погані контакти. У патч-записі струми, що йдуть через контакт, також проходять через вимірювальний контур, збільшуючи шум від вимірюваного струму [14].

Висновок до першого розділу

В електрофізіології клітину та обладнання зручно розглядати як електричну еквівалентну схему, тобто комбінацію резисторів, конденсаторів та батарей [16]. Такий підхід допомагає зрозуміти процеси та автоматизувати процес обробки експериментальних даних.

Узагальнюючи вищезгадане дослід поетапно можна представити наступним чином. Підсилювач можна представити джерелом напруги E_{pc}, послідовно з резистором R_{pc} і ємністю C_{pc} на вході. Піпетка вимірювального приладу може бути представлена опором піпетки R_{pip} і ємністю піпетки C_{pip} коли піпетка потрапляє в розчин. Контакт піпетки з мембраною клітини може бути представлений опором ущільнення R_{seal}. На рис. 1.8 а після формування контакту отвір піпетки закривається (CAP - cell attached pipette) з високим опором R_{cap}. Розрив мембрани замінює R_{cap} на опір доступу R_{acc} (далі R_a), забезпечуючи доступ до внутрішньої частини клітини (WC – whole cell), з параметрами її мембрани E_m, R_m, C_m. На рис. 1.2b видно, що три етапи процедури отримання WC-конфігурації можуть бути замінені простою схемою ERC з трьома перемикачами (S). Замикання S_{pip} (подвійний перемикач з S_{cpip} і S_{rpip}) означало б вхід у ванну піпеткою, відкриття вимикача S_{seal} символізує різку герметизацію, а закриття перемикача S_{acc} еквівалентно встановленню WC шляхом короткого замикання R_{cap} з опором доступу R_{acc} ($R_{cap>>}R_{acc}$) [15].

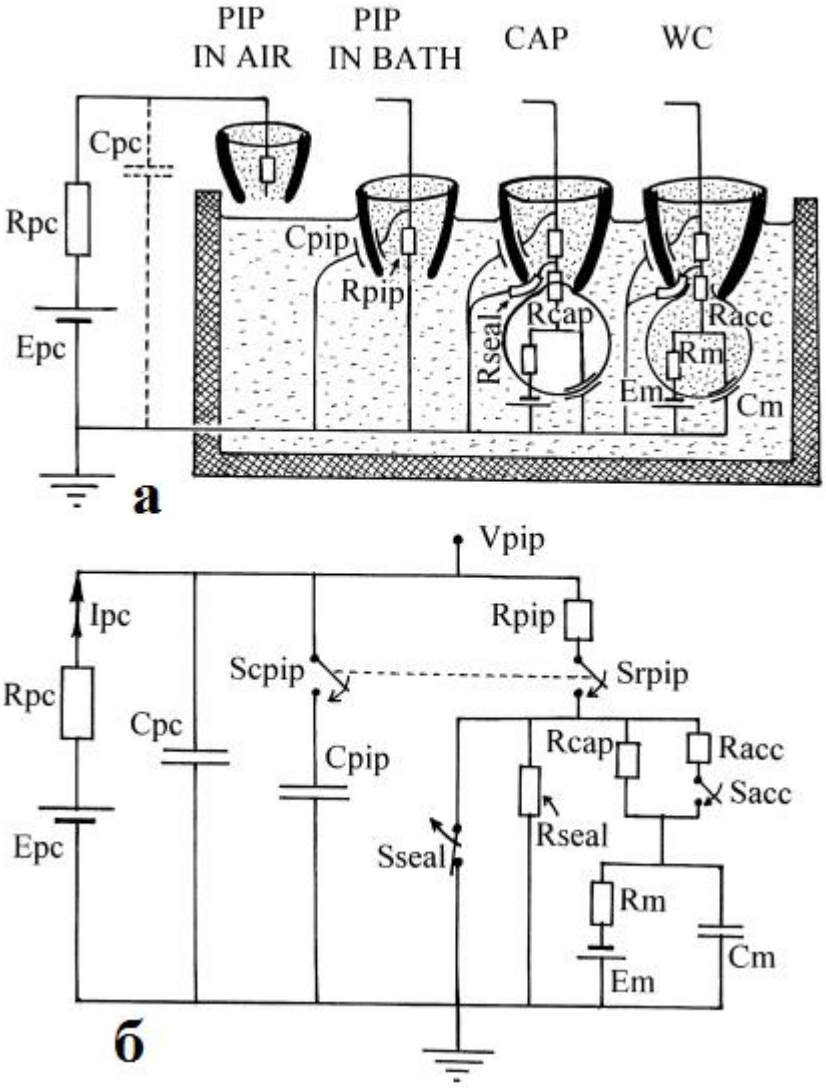


Рисунок 1.8. Простий електричний ланцюг, що моделює послідовні процедури patch-clamp для отримання цілоклітинної конфігурації (WC) [15].

Піпетка (PIP), що входить у ванну, утворює контакт разом з кліткою і розриває її. Частина 1.8а показує, як різні компоненти схеми можна ототожнювати з компонентами електротехніки. Частина 1.8б показує еквівалентну схему для трьох послідовних процедур, що ведуть до конфігурації WC. Під час експерименту якість контакту піпетки та інші параметри визначаються шляхом зміни напруги на E_{pc} та вимірювання отриманого струму I_{pc} [15]. Подальші розрахунки буде розглянуто в третьому розділі.

РОЗДІЛ 2. ОГЛЯД ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ

2.1. Формат фійлів електрофізіологічних даних

Формат бінарного файлу Axon Binary File (ABF) був створений для зберігання бінарних файлів експериментальних даних. Він використовується з пакетом програм pCLAMP (програмний пакет для збору та аналізу електрофізіологічних даних: контролю та запису напруги, струму; складається з програмного забезпечення для збору даних Clampex 11, програмного забезпечення AxoScope 11 для фонового запису, програмного забезпечення Clampfit 11 для аналізу даних та додаткового модуля розширеного аналізу Clampfit для складного та спрощеного аналізу) [43], але також підтримується AxoTape та AxoScope. Ці файли можна створювати та читати на комп’ютерах під Microsoft Windows [5][6].

2.1.1. Структура файлу ABF

Axon Binary File має патентований формат, проте файли можна читати і створювати сторонніми розробниками за допомогою бібліотеки ABFFIO.DLL (динамічно пов'язана Windows бібліотека для доступу до даних) [5][6].

Файл ABF складається з кількох розділів наступним чином:

- Розділ заголовка ABF,
- Розділ налаштування сфери застосування ABF [29],
- Розділ даних ABF,
- Розділ синхронізації ABF,
- Розділ тегів ABF,
- Розділ дельти (відмінностей) ABF [31],
- Розділ даних про форму хвилі стимулу.

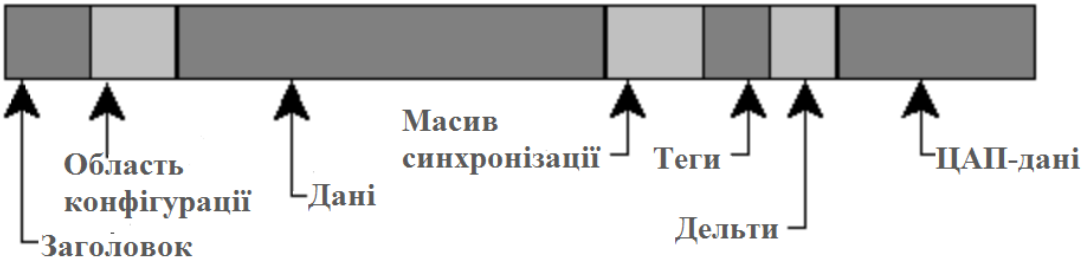


Рисунок 2.1. Структура файлу ABF [5].

Заголовки та розділи даних відображаються у наведеному порядку. Інші розділи можуть з'являються в будь-якому порядку, оскільки на них вказують параметри в заголовку. Усі секції зібрані в блоки по 512 байти кожен. Початкове розташування секції задається як номер блоку. Блок номер 0 являє собою початок файлу. Якщо покажчик номера на розділ (крім заголовка) дорівнює нулю, відповідний розділ не пишеться.

Версія ABF 2 (випущена разом з pCLAMP 10 у 2006 р.) є важливим оновленням попередньої версії ABF. Основна зміна полягає в тому, що заголовок файлу тепер змінної довжини. Це вплинуло на те, що неможливо більше читати дані (або інші розділи) безпосередньо з файлу, а потрібно зробити за допомогою бібліотеки ABFFIO.DLL [5] [6].

2.2. Короткий огляд існуючих обробних комплексів

Програмним забезпеченням для обробки наукових даних, що працює з форматом ABF є Clampfit, MiniAnalisys та Origin.

Clampfit (програмний пакет Axon pCLAMP) від Molecular Devices Corporation для аналізу даних включає в себе широкий спектр процедур фільтрації та фітингу (підгонки): графіки вольт-амперної характеристики, спектри потужності та режими виявлення та аналізу подій [27]. Шуми можна відфільтровувати за допомогою високочастотних, низькочастотних та смугових фільтрів Бесселя, Баттерворта, Чебишева, Гаусса та ін. Існує кілька

різних методів регулювання базових ліній записів: постійні значення або середні показники можна відняти від кожної з точок запису, базові лінії можна відрегулювати, застосовуючи корекцію нахилу, або ручну корекцію за допомогою полілінії, для нестабільних базових ліній. Додатковими функціями аналізу даних є усереднення, нормалізація та вирівнювання піків. Для фітингу (підгонки) кривої є близько сорока попередньо визначених функцій [11].

Mini Analisys від Synaptosft виявляє і аналізує піки різного типу, форми, напрямку та розміру завдяки багаторазовому алгоритму виявлення піків. Це необхідно, наприклад, для спонтанних синаптичних подій. Програма має цифровий фільтр та засоби частотного спектрального аналізу, автоматично коригує базову лінію близько розташованих піків методом експоненціальної екстраполяції розпаду. Обчислює статистичні параметри, будує частотні гістограми, накопичувальні гістограми , діаграми розсіювання та графіки [48].

Origin - пакет програм від OriginLab Corporation для чисельного аналізу даних і наукової графіки [40]. Має безліч засобів для побудови і модифікації діаграм, двовимірних та тривимірних графіків, працює з більшістю форматами та базами даних. Виконує статистичний аналіз (описова статистика, тести гіпотез, аналіз дисперсії та ін. [49]), шукає та аналізує піки, визначає базову лінію, виконує фітинг (підгонку), фільтрування та згладжування багатьма методами, інтра/екстраполяцію, нормалізацію даних та багато іншого [41].

Ці програми є пропрієтарними та мають закритий програмний код [18], тому немає можливості додати деякі функції та параметри обробки (пакет Origin надає таку можливість, в останніх версіях також на Python [41]). Ліцензовані програми для обробки працюють під ОС Microsoft Windows [28] [48] [40]. Це обмежує науковців що користуються іншими операційними системами. Обробка даних виконується окремо для кожного файлу даних (окрім Clampfit 11.1 [28] [11] і останніх версій Origin [41]). Це виключає можливість здійснювати однакові обчислення для даних декількох експериментів, не витрачаючи час на багаторазове введення параметрів

обробки. Вищезазначені програми (окрім Origin [40]) не дають можливості зберігати отримані графіки у векторному форматі, а тому використання цих зображень, наприклад, для плакатів недоречно.

2.3. Вбудована система для patch-clamp

Конфігурації Voltage-Clamp (режим фіксації потенціалу) та Current-Clamp (фіксації струму) є стандартними конфігураціями клітинної електрофізіології. Використовується також третя конфігурація - Dynamic-Clamp (режим фіксації провідності). Робота в цій конфігурації вимагає щоб електрофізіологічна система контролювала мембранний потенціал і використовувала його в режимі реального часу для обчислення які з модельованих іонних каналів були б відкриті, якби вони були фізично присутні.

Ідея пристрою dynamic-clamp проста, на відміну від її реалізації, оскільки динамічні обчислення потрібно робити в режимі реального часу, тобто набагато швидше, ніж будь-які значущі зміни властивостей мембранних каналів або мембранного потенціалу.

Тут описано реалізацію dynamic-clamp пристрою на основі Teensy 3.6. Він може бути використаний для надання можливостей dynamic-clamp до будь-якої готової електрофізіологічної установки без необхідності її перебудови [7].

2.3.1. Метод dynamic-clamp

Dynamic-clamp - це метод електрофізіології, який використовує інтерфейс у реальному часі між однією або кількома живими клітинами та комп'ютером чи аналоговим пристроєм для імітації динамічних процесів, таких як мембранні чи синаптичні струми в живих клітинах.

Принцип роботи dynamic clamp проілюстрований на рисунку 2.2 для

однієї (справа) або декількох (зліва) контактуючих клітин. Їх мембранний потенціал V (V_1 та V_2) підсилюється і подається на пристрій dynamic clamp (сірий прямокутник). Там міститься модель мембрани або синаптичної провідності, яка подається в живу клітину у вигляді рівнянь (для цифрових систем), або відповідним електричним контуром (для аналогових систем). Система dynamic clamp обчислює струм I (I_1 та I_2), що генерується змодельованою мембранною провідністю і виводить її в режимі реального часу. Для оптимальної роботи циклу зчитування потенціалу мембрани, обчислення та генерування відповідного струму, потрібно щоб швидкість оновлення даних була більше, ніж швидкість всіх інших компонентів системи.

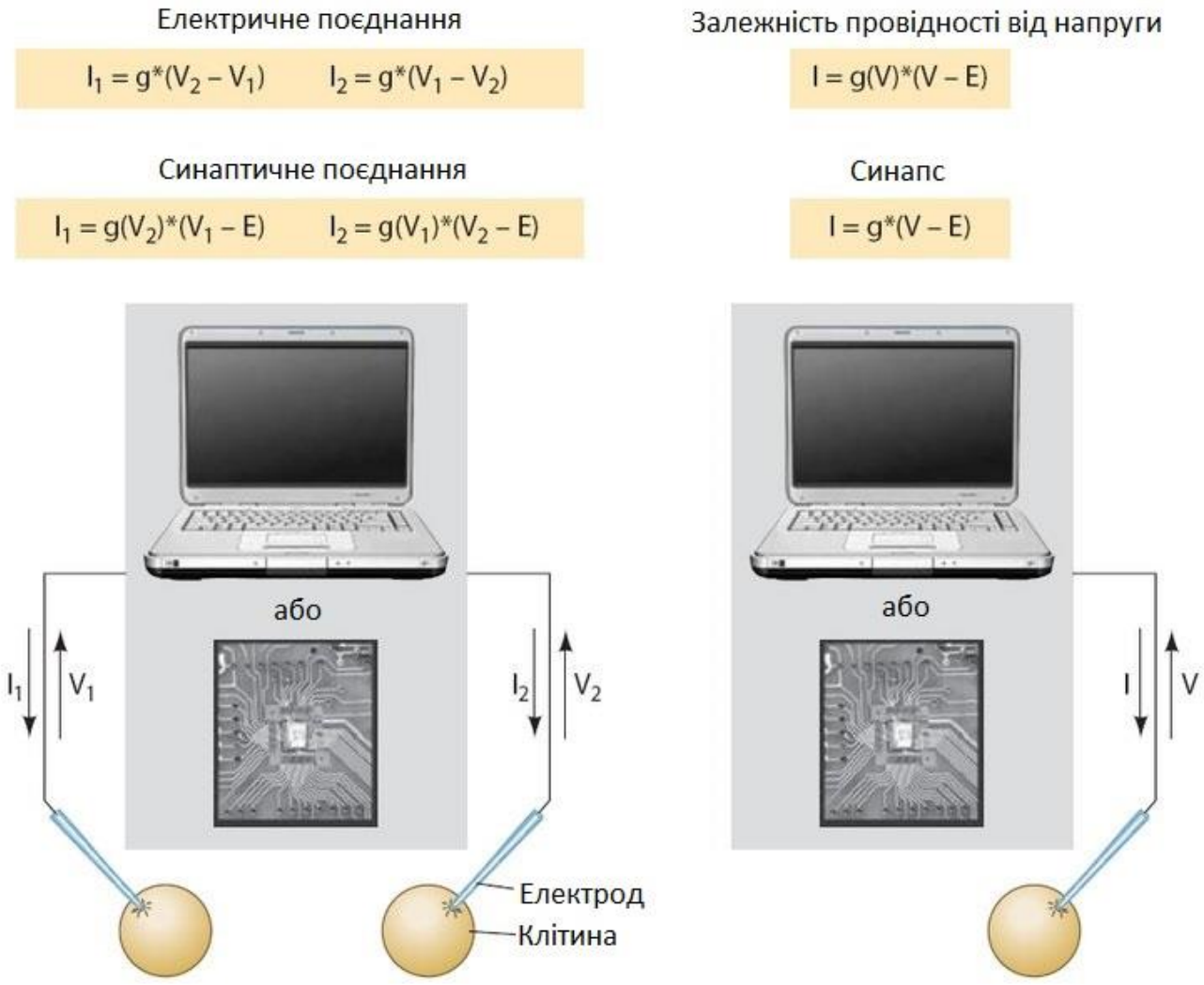


Рисунок 2.2. Принцип роботи dynamic clamp [10].

Залежно від того, які типи електропровідності моделюються за допомогою dynamic-clamp, програми поділяються на одну або кілька категорій [10].

- Додавання або віднімання мембранних струмів

Для додавання або віднімання мембранних струмів спочатку необхідно розробити модель, що описує залежність напруги та часу від заданого мембранного струму. Як тільки модель готова, додавання або віднімання мембранного струму виконується системою dynamic clamp. Деякі поширені формалізми, які використовуються для представлення мембранних струмів, - це моделі Ходжкіна-Хакслі (математична модель, яка описує розповсюдження потенціалів дії в нейронах; являє собою комплекс ординарних диференційних рівнянь, який змальовує характеристики електричного сигналу [17]), ФітцХ'ю — Нагумо (математична модель, що описує прототип збуджуваної системи [20]) та Маркова (імовірнісна модель, що використовується для моделювання систем які змінюються випадковим чином [38]). Для моделювання з допомогою системи dynamic clamp використовують модель, створену наближенням її до експериментальних даних мембранного струму [10].

- Додавання або виключення синаптичних з'єднань

Система може бути використана для додавання або виключення електричних щілинних контактів та хімічного збуджуючого або гальмівного синаптичного входу між двома або більше клітинами. В одному із варіантів реєструється пресинаптичний нейрон і його генерацію потенціалів дії контролює штучний синапс, підключений до нейрону постсинаптичного типу. Аналогічно, ефект існуючого синаптичного з'єднання можна виключити, додавши негативну провідність, на противагу потенціалу що виник. Така конфігурація використовується для дослідження впливу часу введення синаптичного сигналу на баланс збудливого/гальмівного постсинаптичного сигналу реальних нейронів [10].

- Моделювання мереж

Система може бути використана також для побудови гібридних мереж реальних і змодельованих нейронів з метою систематичної ізоляції та дослідження їх властивостей, щоб краще зрозуміти поведінку мережі нейронів [10].

2.3.2. Застосування і обмеження методу

Dynamic clamp може бути використаний для імітації дії фармакологічних препаратів, оскільки введення або виключення струму (в іонному каналі або синапсі) може мати такий ефект, як і активація або інактивація струму препаратами. Також це винятковий практичний навчальний інструмент. Замінюючи реальний нейрон змодельованим нейроном, усе обладнання для реального експерименту може бути використане для викладання концепцій електрофізіології.

Час, необхідний для зчитування мембранного потенціалу та обчислення струму для введення, повинен бути менше, ніж найшвидша константа часу (що характеризує час заряджання і розрядження мембрани) в реальному нейроні. З постійно зростаючими швидкостями комп'ютерних процесорів обмеження швидкості в dynamic clamp постійно відштовхується. Інтервал оновлення повинен бути сталим. Сучасні операційні системи як Linux, Mac OS та Windows, не дають гарантії для програми, що працює в ОС, виконання операції саме тоді, коли прийде запит. Linux з розширенням ядра RTLinux, LabView з модулем реального часу та DOS може гарантувати виконання у чітко визначений часовий проміжок. Ще одне обмеження - відстань. Система вводить струм обмежений простором навколо записуючого електрода. Важливо зазначити, що неелектричні ефекти не імітуються dynamic clamp. Хоча відсутність хімічних ефектів часто обмежує фізіологічний реалізм модельованих струмів, це може бути корисно, якщо електричні та хімічні ефекти одного і того ж білка мембрани вивчаються нарізно [10].

2.3.3. Доступні на сьогодні системи динамічної фіксації

Нижче наведено перелік існуючих та доступних систем, він не є вичерпним:

- на основі Windows пакет програмного забезпечення зі зручним графічним інтерфейсом;
- на базі Windows з можливістю контакту з кількістю нейронів до чотирьох одночасно і новий варіант тієї ж системи - StdpC, що дозволяє моделювати гнучку, залежну від часу потенціалу дії штучних хімічних синапсів компенсацію активного електрода для одноелектродної конфігурації;
- NetClamp вирішує взаємопов'язані цілі моделювання нейронних мереж та проведення динамічних експериментів, що дозволяє будувати мережі з різною кількістю змодельованих та реальних клітин з різними синаптичними зв’язками між ними. Керується за допомогою карт збору даних National Instruments;
- QuB - це динамічна система, орієнтована на кінетичне моделювання іонних каналів з напругою, що працює під управлінням Microsoft Windows в реальному часі, використовуючи паралельну обробку на багатоядерних або багатопроцесорних машинах;
- G-clamp - система, заснована на LabView в реальному часі;
- RTXI - система в режимі реального часу на базі Linux, що виникла в результаті злиття NSF і трьох попередніх систем;
- платформа даних ITC-16 дозволяє реалізувати штучну синаптичну провідність на основі програмованих масивів іонних воріт;
- Signal - це комерційно доступна програма збору та аналізу даних загального призначення, яка забезпечує високу ефективність (до 100 КГц) завдяки швидкому процесору RISC, вбудованого в апаратне забезпечення збору даних Power1401;
- National Instruments (NI PXI-8176), керована Labview за допомогою модуля реального часу;

- цифрова плата обробки сигналу (DS1104 ; DSPACE, Novi, MI), керована Matlab з Simulink;
- динамічна система підключення та відтворення від Cytocybernetics;
- мікроконтроллер Teensy 3.6, подібний до Arduino, може використовуватися для додавання функцій до існуючих електрофізіологічних систем (Windows, Macintosh та Linux) [4].

Останню розглянемо детальніше.

2.3.4. Пристрій для емуляції вхідних синаптичних струмів

На рисунку 2.3 елементи, присутні на кожній внутрішньоклітинній електрофізіологічній установці: підсилювач (Amplifier) та плати збору даних (data acquisition, DAQ). Підсилювач може бути: Multiclamp 700B, Dagan BVC-700A, HEKA EPC-10, AM Systems 2400, або будь-який інший, який контролює мембранний потенціал нейрона і вводить струм в цей нейрон за допомогою мікропіпетки чи мікроелектроду. В якості DAQ можуть бути Molecular Devices Digidata 1500, HEKA ITC-18, National Instruments PCIe-6343, або будь-яка інша з величезної кількість плат, які працюють з операційними системами Macintosh, Windows або Linux. Плата DAQ управляється на хост-комп'ютері системою DAQ, придатною для внутрішньоклітинної електрофізіології. Ця система може бути однією з комерційних систем (наприклад Molecular Devices pClamp 10 або AxoGraph), системою з відкритим кодом (наприклад Janelia's Wavesurfer), або бути саморобною.

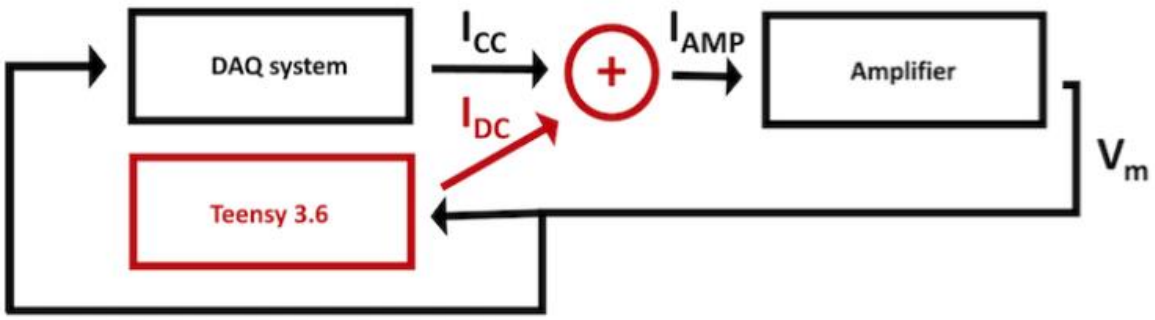


Рисунок 1.3. Електрофізіологічна установка з доповненням [7].

До існуючої робочої конфігурації додається частина, що складається з мікроконтролера Teensy 3.6 та відповідних електронних компонентів, що забезпечують взаємодію мікроконтролера з підсилювачем та DAQ. Цей мікроконтролер порівняно з іншими пристроями цього класу та у його ціновому діапазоні досить швидкий (тактова частота 180 МГц), має значну пам'ять (256 кБ оперативної пам'яті) та здатен рахувати числа з плаваючою крапкою. Teensy відповідає за виконання всіх динамічних розрахунків. Він миттєво визначає, який струм буде проходити через мембранний канал, якби він був фізично присутній, і додає це до струму, який система DAQ повинна пропускати через клітину.

Для роботи Teensy потрібні деякі доповнення (рис. 2.4): (1) - блок живлення; (2) - схема для перетворення напруги внутрішньоклітинного підсилювача, що представляє мембранний потенціал нейрона, в межах ± 9 В, у межі, які може прочитати Teensy (0-3,3 В); (3) - електричні контакти входів та виходів контролера; (4) - схема перетворення, вихідної напруги від Teensy 0-3,3 В у напругу ± 9 В, яку підсилювач може правильно інтерпретувати та дати відповідь у вигляді струму (в pA), який повинен бути введений в нейрон; (5) - ланцюг для сумації струмів від модулю dynamic-clamp і струмів, що продукуються системою DAQ.

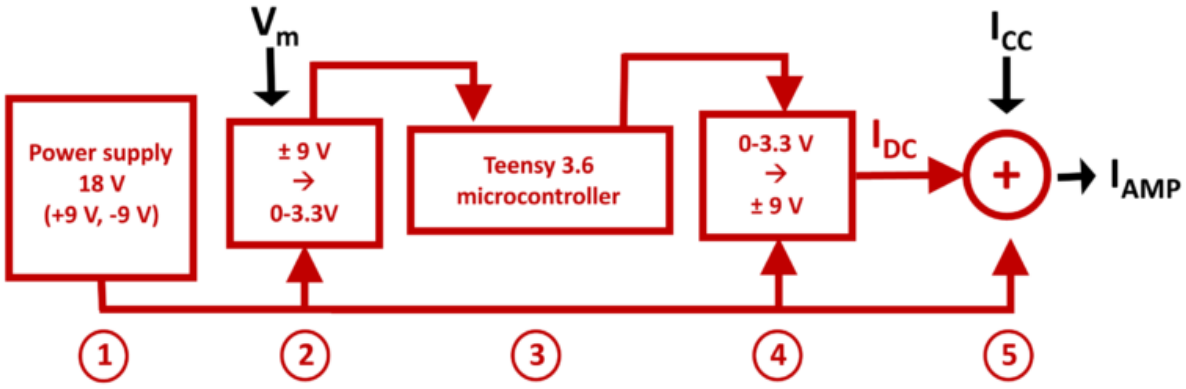


Рисунок 2.4. Структура системи на основі мікроконтролера Teensy 3.6 [7].

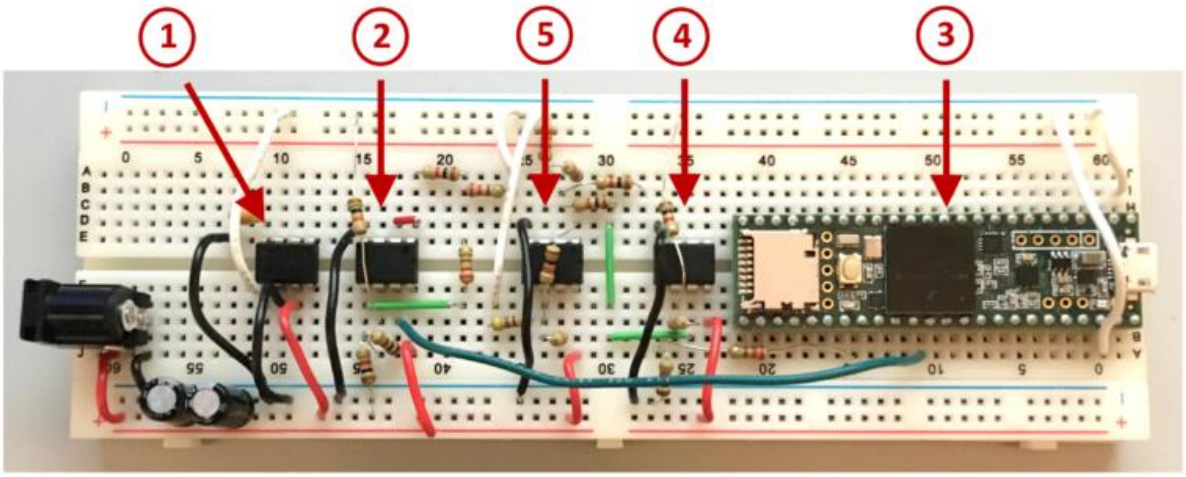


Рисунок 2.5. Приклад реалізації пристрою на контактній платі без використання пайки [7].

Хоча Teensy 3.6 не є мікроконтролером лінії Arduino, він дуже схожий і може бути запрограмований за допомогою відкритого інтегрованого середовища розробки Arduino (IDE). Також виконується код, написаний на мові Processing (open source), щоб оброблювати параметри під час запису. Processing є дуже простою мовою і, як і Arduino, код можна використовувати без змін у розповсюджених операційних системах (Macintosh, Windows, Linux). Весь вихідний код проекту опубліковано на рипозиторії Github.

Існує кілька способів програмування Teensy 3.6, включаючи просто використання мови C. Але найпростіший спосіб зробити це - через Arduino IDE. Програмний комплекс Arduino IDE і пов'язана з ним мова програмування зберігають основи синтаксису C та одночасно спрощують процес взаємодії з мікроконтролером. Хоча Teensy не є частиною лінійки мікроконтролерів Arduino, існує плагін до Arduino IDE для програмування Teensy, який дозволяє використовувати IDE та більшість його бібліотек.

Програмне забезпечення для модулю dynamic-clamp з використанням Teensy 3.6 написане за допомогою Arduino IDE. Відповідний код міститься в вищезгаданому репозиторії GitHub в директорії dynamic_clamp, а головний файл називається dynamic_clamp.ino. Крім нього в зазначеній директорії лежать пов’язані з ним файли. Кожен файл містить код, що моделює

певну іонну провідність. Головний файл складається з трьох частин: оголошення глобальних змінних, функції setup() та функції loop(). Функція setup() відповідає за початкове налаштування пристрою (ініціалізацію), та запускається один раз, коли програма завантажується на плату і робить такі дії, як ініціалізація послідовного зв’язку між хост-комп'ютером та платою Teensy; функція loop() запускається на кожному часовому кроці - вона викликає кожен із файлів, що описують моделі іонної провідності, щоб отримати струм, визначений цією провідністю. Моделі електропровідності у відповідних файлах: Shunting.ino, HCN.ino, Sodium.ino, EPSC.ino - ці чотири приклади охоплюють основний діапазон електропровідностей. Представлений на GitHub код підходить у якості шаблонів, за якими можна прописати власні моделі електропровідностей [7].

Коли програма Arduino завантажується на мікроконтролер Teensy, всі динамічні моделі ініціалізуються як нульові. Вони можуть бути змінені на ненульові значення під час роботи програми.

Найпростіший спосіб для керуючого комп'ютера повідомити мікроконтролеру змінити динамічну модель провідності - через порт USB, який їх з'єднує. Arduino мікроконтролер постійно перевіряє зв’язок з хост-комп'ютером та змінює значення провідності, як тільки воно надходить. Сам Arduino IDE не має хорошого способу передачі команд в реальному часі з хост-комп'ютера на мікроконтролер. На щастя, існує багато інших програм, наприклад, програмне рішення під назвою Processing підходить для цієї мети: це середовище з відкритим кодом та з простим синтаксисом (заснований на Java і, таким чином, має схожість з сімейством з С-подібних мов). Він має велику базу користувачів та не залежить від платформи.

У директорії processing_control знаходиться прототип Processing (файл processing_control.pde), який створює графічний інтерфейс користувача (GUI), за допомогою якого можна змінювати значення максимальних коефіцієнтів провідності для чотирьох провідностей, описаних вище, наявних за

замовчуванням. GUI також дозволяє користувачам змінювати константи дифузії для представлених моделей. Всі параметри можна змінити, переміщуючи повзунки в графічному інтерфейсі та натискаючи кнопку «upload». Але це не єдиний можливий спосіб взаємодії з мікроконтролером у реальному часі. Користувач може використовувати будь-яке програмне забезпечення, яке він бажає, щоб, наприклад, більш щільно поєднати програмне забезпечення для отримання даних та управління dynamic-patch у реальному часі.

Висновок до другого розділу

Існує безліч конфігурацій експериментів, для кожного з них є досить багато програмних рішень, що мають схоже застосування. Зазвичай кожне програмне рішення має певні обмеження в застосуван, наприклад, підходить лише для певного набору обладнання та конфігурації експерименту. Або навпаки, вони досить універсальні, включають в себе безліч функцій, але в них немає необхідних вузькоспецифічних. Іноді для них необхідні додаткові розширення, або створення макросів. У випадку, коли для конкретного дослідження функціоналу не існує в жодній із представлених на ринку програм, це потребує створення власних. В наступному розділі описано створення саме такої системи для більш специфічних потреб.

РОЗДІЛ 3. АНАЛІЗ РОЗРАХУНКОВИХ ПАРАМЕТРІВ ТА РОЗРОБКА СИСТЕМИ

Розлянемо основні параметри, що потребують обробки. Визначимо їх фізичний зміст та математичну модель. Встановимо зв'язок між параметрами та торкнемося питання похибки аналізу даних та варіанти розв’язання проблеми.

З усіх електрофізіологічних параметрів, для розв’язання задачі оцінки якості експериментальних даних, візуалізації отриманих даних та покращення якості, виділимо основні параметри контакту з клітиною – опір доступу та струм витоку, а також пасивні параметри клітини.

Частково проаналізуємо принцип і користь фільтрації даних, негативні наслідки застосування фільтрів.

3.1. Пасивні параметри.

3.1.1. Опір мембрани, R_M

Клітинна мембрана складається з подвійного ліпідного шару, який відокремлює іони у позаклітинному просторі від іонів та заряджених білків у внутрішньоклітинному просторі. Хоча чисті ліпідні мембрани є абсолютними електричними ізоляторами, справжні клітинні мембрани складаються з щільної мозаїки білків і ліпідів. Багато з цих білків пронизують мембрану і діють як канали, що дозволяють проходити заряду, працюючи як провідники. Ці білки знижують високий опір мембрани, що має значні наслідки. Припустимо, необхідно хочемо прикласти напругу на клітинну мембрану шляхом введення струму електродом. Струм, необхідний для підтримання цієї напруги, визначається мембранним опором, згідно із законом Ома: $V=R\cdot I$. Тому, чим більший опір мембрани, тим менший струм необхідний для підтримання заданої мембранної напруги [16].

3.1.2. Ємність мембрани, C_M

Оскільки мембрана є електричним ізолятором, що розділяє протилежні заряди всередині і зовні клітини, то мембрану характеризує не тільки опір, але і ємність. Тому для зміни напруги на мембрані необхідно зарядити ємність. Прикладений заряд Q, розділений на ємність мембрани C_M, дає напругу мембрани V_M:

$$V_M = \frac{Q}{C_M} \ . \qquad (3.1)$$

З (3.1) бачимо, що при заданій кількості прикладеного заряду, чим менша ємність мембрани, тим більша зміна напруги мембрани [16].

3.1.3. Поєднання R_M і C_M - ланцюг RC

Оскільки мембранний опір R_M та мембранна ємність C_M одночасно виникають у клітинній мембрані, вони електрично паралельні (рис. 2.1, А). Така схема паралельного опору R та ємності C відома як RC-ланцюг. RC-ланцюги зазвичай використовуються в електроніці як основні фільтри для вибору діапазонів вхідних частот. Аналогічно клітинна мембрана діє як фільтр струму або напруги, що вводиться в клітину [16].

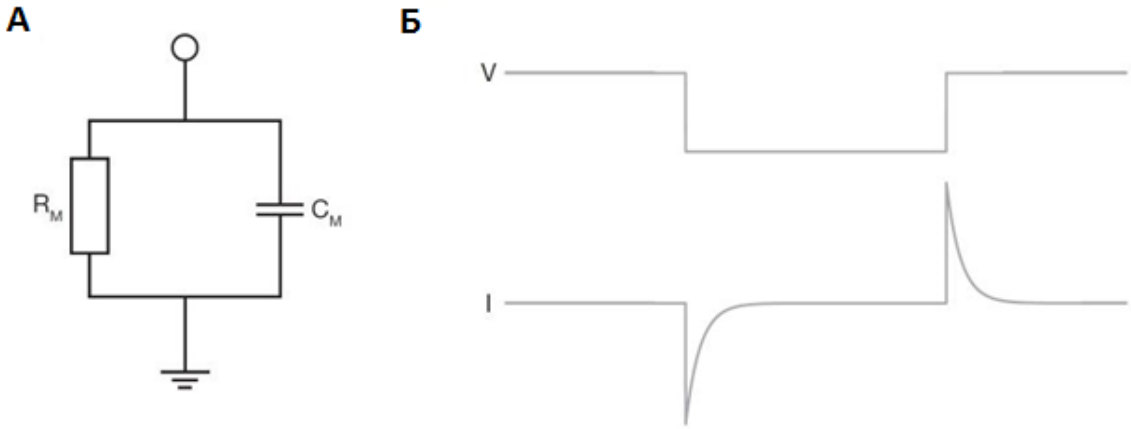


Рисунок 3.1. Основні схеми електричних властивостей плазматичної мембрани. А - схема, що показує мембранну ємність і опір мембрани паралельно один одному.Б - графіки, що показують прикладений до клітини імпульс напруги (вгорі) та результуючу реакцію струму (знизу) для простої плазматичної мембрани, в режимі фіксації напруги (voltage-clamp) [16].

3.2. Параметри контакту

3.2.1. Параметри базового експерименту у режимі фіксації потенціалу

Щоб зрозуміти, як властивості RC-фільтра мембрани визначають відповідь напруги на мембрані клітини. Розглянемо, як крок напруги, прикладений до клітини, змінює струм, що вводиться через електрод (рис. 3.1, Б). Спочатку крок напруги квадратної форми призводить до миттєвого стрибка струму (початковий пік). Потім цей струм збільшується експоненціально (зростаюча ділянка) до досягнення стійкого стану. Навпаки, коли напруга повертається у вихідне значення, ми спостерігаємо стрибок ємнісного струму протилежного напрямку, який зменшується експоненціально, поки він знову не досягне сталого стану. Контроль напруги мембрани та вимірювання отриманого струму таким чином є основним експериментом у режимі фіксації потенціалу voltage-clamp.

Спочатку весь струм заряджає мембранну ємність (рис. 3.1, Б), а струм, що проходить через опір мембрани, відсутній. Таким чином, амплітуда початкового швидкого струму цілком визначається величиною кроку поданої напруги та опором електрода (що визначається як сума власного опору електрода та опору контакту електрода з клітиною). Коли ємність мембрани (membrane capasistance, C_M) стає все більш зарядженою, все більша частка введеного струму протікає через опір мембрани. Після того, як ємність мембрани C_M повністю заряджена, система досягає стійкого стану і весь струм протікає через опір мембрани (membrane resistance, R_M). У стаціонарному стані величина струму, необхідна для підтримки напруги мембрани (стаціонарний струм), залежить лише від мембранного опору і визначається законом Ома - часткою від тестуючого кроку напруги V_T (поданий на мембрану імпульс напруги) і опору мембрани R_M:

$$I_S = \frac{V_T}{R_M} \qquad (3.2).$$

Значення мембранної ємності та мембранного опору визначають, наскільки швидко досягається стаціонарний стан: чим більша ємність чи опір, тим довше емність мембрани буде заряджатись. Постійна часу τ, с, що описує цей процес, відома як константа часу мембрани і дорівнює добутку мембранного опору R_M та мембранної ємності C_M (1.7) [16].

Зазначені вище взаємозв'язки використовуються для моніторингу характеристик клітини на різних етапах раєстрації. Для цього прикладають невеликий імпульс напруги на електроді, так званий тестовий імпульс, і спостерігають за формою та амплітудою поточної відповіді (рис. 2.1, Б).

3.2.2. Конфігурація цілої клітини (whole-cell)

Після утворення щільного контакту клітини з піпеткою, розривом мембрани під кінчиком піпетки отримують електричний доступ до цитоплазми клітини, зберігаючи щільний контакт. Така конфігурація запису називається "ціла клітина" (whole-cell). У цій конфігурації піпетка електрично

безпосередньо з'єднана з клітиною: електрод може «бачити» електричну активність всередині клітини.

Струм між записуючим електродом і заземленням тепер може надходити всередину і проходити через мембрану усієї клітини. Незначна кількість струму проходить повз контакт клітини з піпеткою, оскільки опір контакту принаймні на порядок більший, ніж мембранний опір (тепер опір мембрани визначається всією мембранною клітини, а не лише ділянкою мембрани всередині наконечнику піпетки). Оскільки мембрана під кінчиком піпетки розривається і не видаляється з її просвіту, компоненти мембрани перешкоджатимуть доступу струму від електрода до клітини і сприятимуть так званому "опору доступу". Сума опору доступу (access resistance, R_A) та власного опору піпетки (R_{pip}) складають загальний опір на кінчику піпетки, який називають послідовним опором (R_{series}) [16].

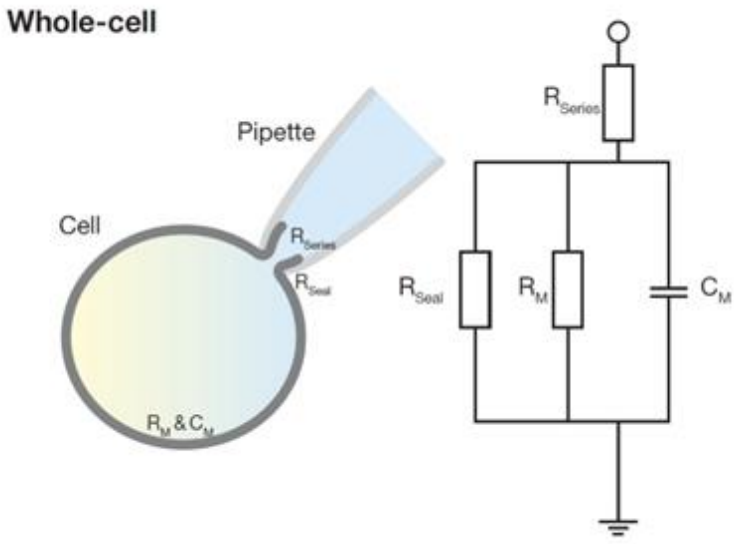


Рисунок 3.2. Схематичне пояснення конфігурації "whole-cell". У цій конфігурації струм визначається послідовним опором піпетки (R_{Series}) послідовно з паралельним контуром опорів мембрани (R_M) та контакту піпетки з мембраною (R_{Seal}), а також мембранною ємністю C_M. Ліворуч вказана піпетка у контакті з клітиною, праворуч відповідна електрична

схема. Реєстрація напруги та струму зображені на рис. 3.1 [16].

3.2.3. Відповідь струму на імпульс напруги у конфігурації «whole cell». Розрахунок опору

Аналізуючи відповідь струму на тестуючий імпульс напруги, можна одразу виділтити два параметри: початкова зміна струму у відповідь на подачу тестуючого імпульсу напруги ΔI та струм утримання (holding current, I_H) (рис. 3.3).

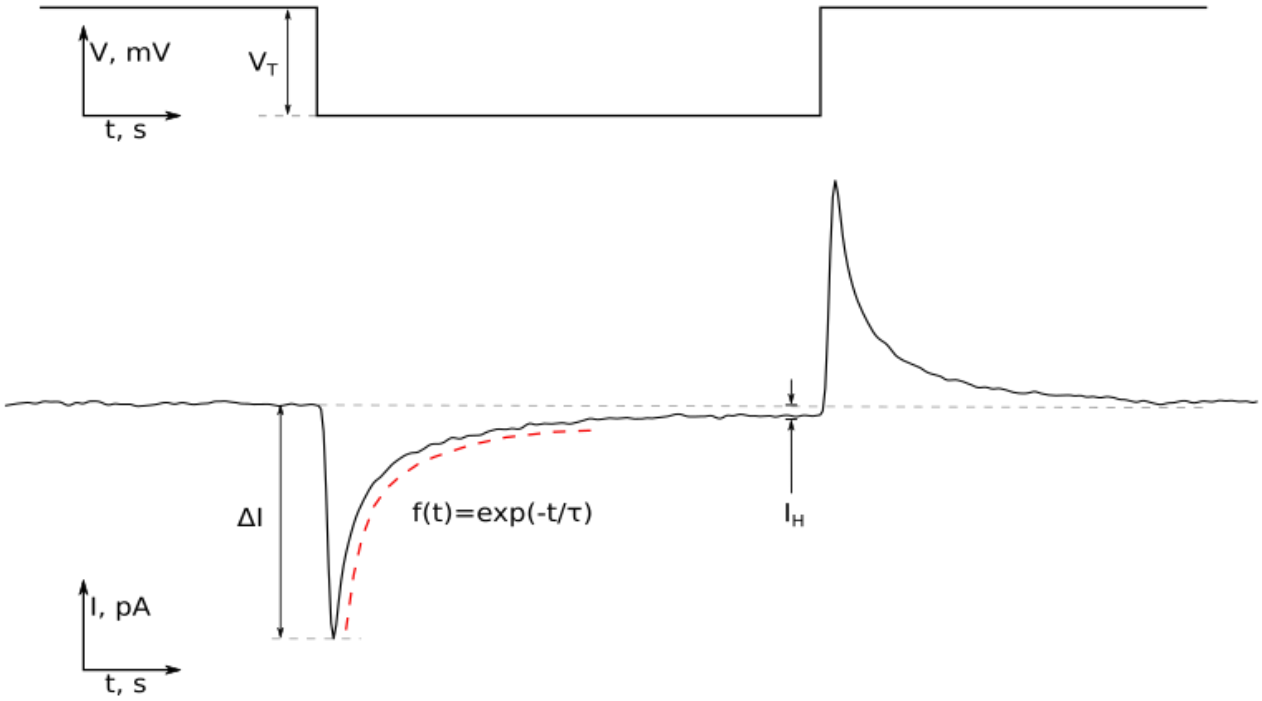


Рисунок 3.3. V_T - тестуючий імпульс напруги; ΔI — амплітуда відповідної зміни струму; I_H — струм утримання; штрихова лінія — експонента, апроксимована до експоненційного спадання струму. По осі абсис відкладено час, по осі ординат — напруга (для верхнього графіку) і сила струму (для нижнього).

Як і раніше, швидкий початковий стрибок струму, так званий транзієнтний ємнісний струм, визначається потоком заряду через піпетку:

$$R_{Series} = \frac{V_T}{I_P}, \qquad (3.3)$$

де R_{Series} - послідовний опір,

V_T - тестуюча напруга,

I_P - початковий струм.

Можна додатково спростити обчислення, використовуючи опір в одиницях МОм (10^6 Ом), напругу в одиницях мВ (10^{-3} В) і струм в одиницях нА (10^{-9} А): $1 \cdot 10^{-3}$ В = $1 \cdot 10^6$ Ом · $1 \cdot 10^{-9}$ А.

Таким чином, якщо у відповідь на імпульс напруги V_T = -5 мВ початковий струм змінюється на -600 пА (рис. 2.4), то послідовний опір становитиме 8,3 МОм:

$$R_{Series} = \frac{-5\ [\text{мВ}]}{-0{,}6\ [\text{нА}]} = 8{,}3\ \text{МОм}.$$

Експоненційне спадання струму визначається постійною часу мембрани τ (1.7), яка також є добутком мембранної ємності та мембранного опору.

$$I(t) = exp\left(\frac{-t}{\tau}\right), \qquad (3.4)$$

де t – час.

$$\tau = R_\text{M} * C_\text{M}\ \forall\ R_{Series} \ll R_M\ . \qquad (3.5)$$

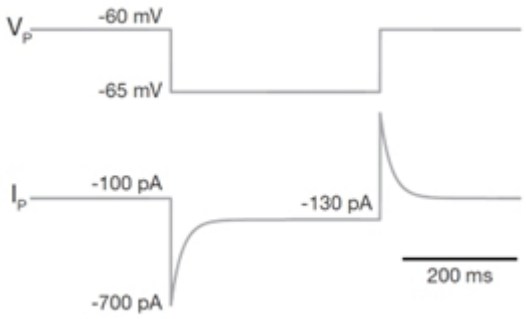


Рисунок 2.4 [16].

Нарешті, як тільки струм досяг стану плато, зміщений струм (струм утримання, I_H) визначається мембранним опором:

$$R_M = \frac{V_T}{I_H}\ . \qquad (3.6)$$

Знайшовши мембранний опір R_M можна обчислити мембранну ємність C_M, визначивши постійну часу мембрани τ, апроксимуючи експоненціальне спадання струму експонентою (3.4):

$$C_M = \frac{\tau}{V_T}\ . \qquad (3.7)$$

Опіру доступу (access resistance, R_A):

$$R_A = \frac{V_T}{\Delta I}, \qquad (3.8)$$

де ΔI — амплітуда ємнісного струму, що виникає при прикладенні до клітини тестового імпульсу напруги (рис. 3.3) [16].

3.2.4. Апроксимація значень

Розрахунок опору доступу лише вимірюванням амплітуди ємнісного струму має суттєвий недолік. Отримане в процесі такого розрахунку значення R_A дуже варіюється в залежності від того, наскільки інтенсивно і за допомогою якого алгоритму відбувалася фільтрація записаного треку. На рисунку 3.5 показано як фільтр Бесселя зміщує початок події під час запису і зменшує амплітуду пікових змін струму при обробці.

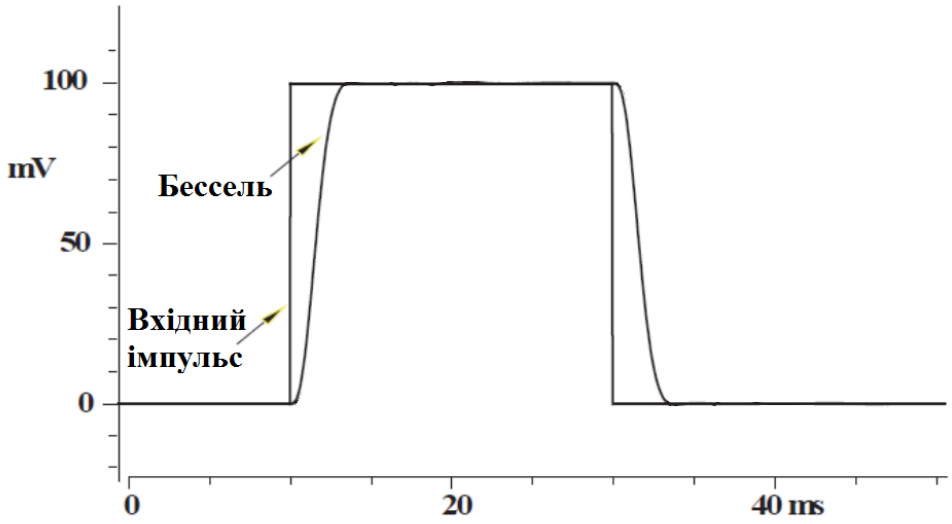


Рисунок 3.5. Зсув що дає фільтр Бесселя четвертого порядку [14].

Так само фільтр Гауса тим сильніше зменшує амплітуду пікової зміни струму ΔI, чим більшим було обрано параметр сигма (σ) при обробці фільтром Гауса (рис 3.6). Через це, в разі роботи з відфільтрованим треком, стає неможливим визначити опір доступу R_A за допомогою вимірювання амплітуди ємнісного струму. Тому доцільніше вираховувати параметр R_A, використовуючи апроксимовану експоненту до спадаючого експоненційного струму. Для цього достатньо знайти точку на апроксимованій експоненті, що

відповідає часу прикладення тестуючого імпульсу напруги і підставити цей час у формулу 13, що описує апроксимовану до струму експоненту. Теоретично вирахуване значення R_A набагато точніше описує реальну величину опору доступу оскільки менше залежить від обраних під час реєстрації струму алгоритмів фільтрації.

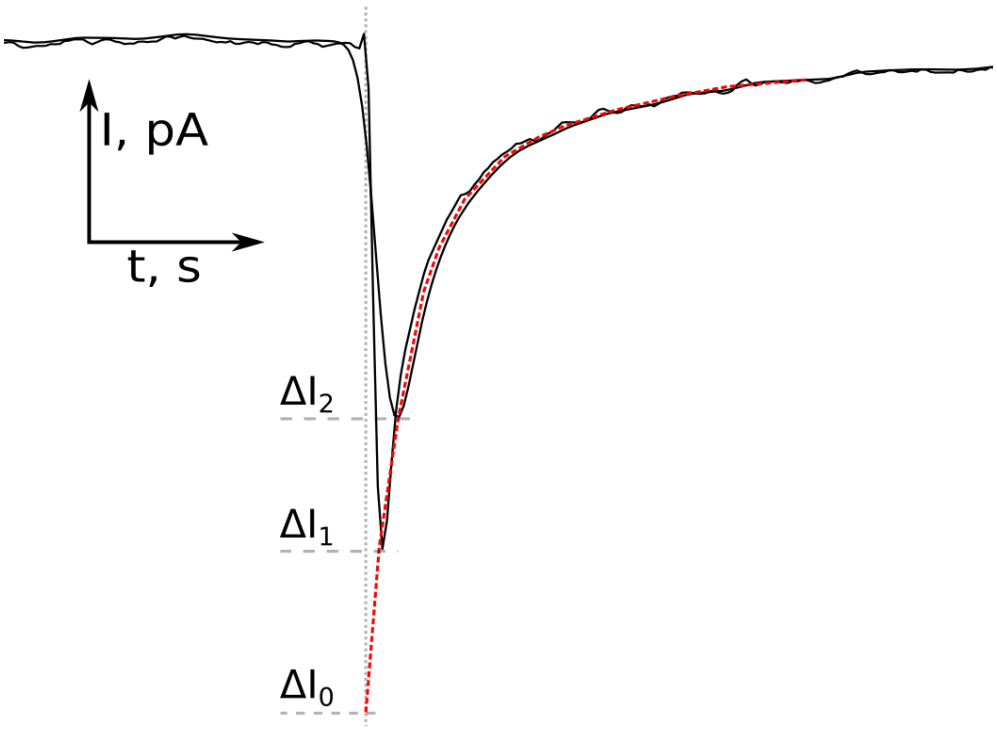


Рисунок 3.6. Відмінність амплітуди ємнісного струму ΔI в залежності від обраної інтенсивності фільтрації по Гаусу. ΔI_1 — фільтрація при $\sigma = 0{,}05$; ΔI_2 — фільтрація при $\sigma = 0{,}15$; ΔI_0 — значення амплітуди ємнісного струму, вирахуване теоретично у точці перетину апроксимованої експоненти (штрихова лінія) з часом прикладення імпульсу тестової напруги (пунктирна лінія). По осі абсис відкладено час, по осі ординат — силу струму.

Далі розглянемо докладніше причини виникнення шуму в реєстраціях.

3.3. Фільтрація шумів та усереднення даних

3.3.1. Шуми

Шуми можуть створювати вимірювальні прилади та обладнання, а також будь-які сторонні прилади. Для вимірювань потрібен низькошумовий підсилювач напруги [14]. Digidata 1320A, який було використано при записі даних, - це низькошумовий оцифровувач, призначений для точного наукового застосування [8].

Сторонні електричні шуми, що не пов’язані з внутрішнім шумом приладу, поділяються на шум від електромережі, магнітно-індукований та шум від поганого заземлення [14]. Основні джерела шуму - це джерела живлення, кабелі живлення, монітор чи будь-які інші. Джерелом змінного магнітного поля може бути трансформатор, електродвигун, наприклад, що знаходиться у вентиляторі чи холодильнику. Високочастотні компоненти шуму (від 20 до 50 кГц) можуть з'являтися за наявності заземлення [8]. Однією з головних проблем в електрофізіології є поява частот 50 або 60 Гц в процесі збору даних.

Лінійно-частотний шум 50/60 Гц, також відомий як електричний гул, що є найпоширенішим джерелом фонового шуму в експериментах patch-clamp [32]. Він виникає від змінного струму електричної мережі [37]. Електричний шум включає шум від освітлення та розетки живлення, високочастотний шум від комп'ютерів. Цей тип шуму зазвичай зменшується, якщо розмістити струмопровідні екрани та використовувати екрановані кабелі. Традиційно для екранування мікроскопа використовують клітку Фарадея. Однак, джерело постійного струму для лампи мікроскопа може мати значну пульсацію змінного струму. Перфузійна трубка, заповнена розчином, що надходить у ванну, може виступати антеною. В такому випадку застосовують екранування труб, або резервуар для подачі крапель, що використовується для внутрішньовенних перфузій (потік рідини переривається і вже не є провідником).

Шум заземлення виникає, коли екранування заземлено в більш ніж одному місці. В цьому випадку струм створює магнітні поля.

Шум від підсилювача та електрода включає тепловий, діелектричний та інші [14].

Традиційно один із способів зменшення шуму - їх екранування. Для усунення шумів лінійних частот використовуються звукові фільтри, або інші методи фільтрації [12].

Підсилювач має вбудовані високочастотні та низькочастотні фільтри [14]. MultiClamp 700B, що був використаний для отримання експерементальних даних, має вбудований 8-полюсний фільтр Бесселя [13]. Однак частину шумів доводиться прибирати при обробці даних. Це виконується за допомогою цифрових фільтрів. Для цієї мети найчастіше використовується фільтр Гаусса через його високу швидкість [14].

3.3.2. Цифрові фільтри. Фільтр Гауса

Цифрові фільтри можна розділити на рекурсивні та нерекурсивні. Вихід нерекурсивного фільтра залежить лише від вхідних даних. Немає залежності від історії попередніх результатів. Приклад - фільтр згладжування на 3:

$$y_n = \frac{x_{n-1}+x_n+x_{n+1}}{3}, \qquad (3.9)$$

де y_n і x_n - вихідні та вхідні зразки на інтервалі вибірки n. Інший приклад нерекурсивного цифрового фільтра - це фільтр Гаусса. Він має схожу форму з описаним вище фільтром згладжування на 3.

Ці типи фільтрів мають перевагу в тому, що вони не змінюють фазу сигналу, на відміну від аналогових, що завжди вносять затримку у відфільтрований сигнал (див. рис. 3.5). Проблема цифрових фільтрів полягає в тому, що значення біля початку та кінця даних не можуть бути належним чином обчислені. Наприклад, у формулі вище, якщо значення є першою точкою даних, x_{n-1} не існує. Це може не позначитися на довгій послідовності точок даних, однак ефект для короткої послідовності може бути відчутним.

Вихід рекурсивного фільтра залежить не тільки від входів, але і від попередніх виходів. Тобто фільтр має деяку залежну від часу «пам'ять». Реалізація цифрових фільтрів Бесселя та Батерворда є рекурсивними [14].

Приблизно 68,26% значень сигналу повинні лежати в межах одного стандартного відхилення σ від μ. Понад 95% значень має бути менше 2σ від μ. І величезні значення повинні бути вкрай рідкісними. На практиці це не так.

Випадкова величина x в нормі розподілена відповідно функції Гауса [3]:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \qquad (3.10)$$

де μ - математичне очікування,

σ- середньоквадратичне відхилення розподілу [23].

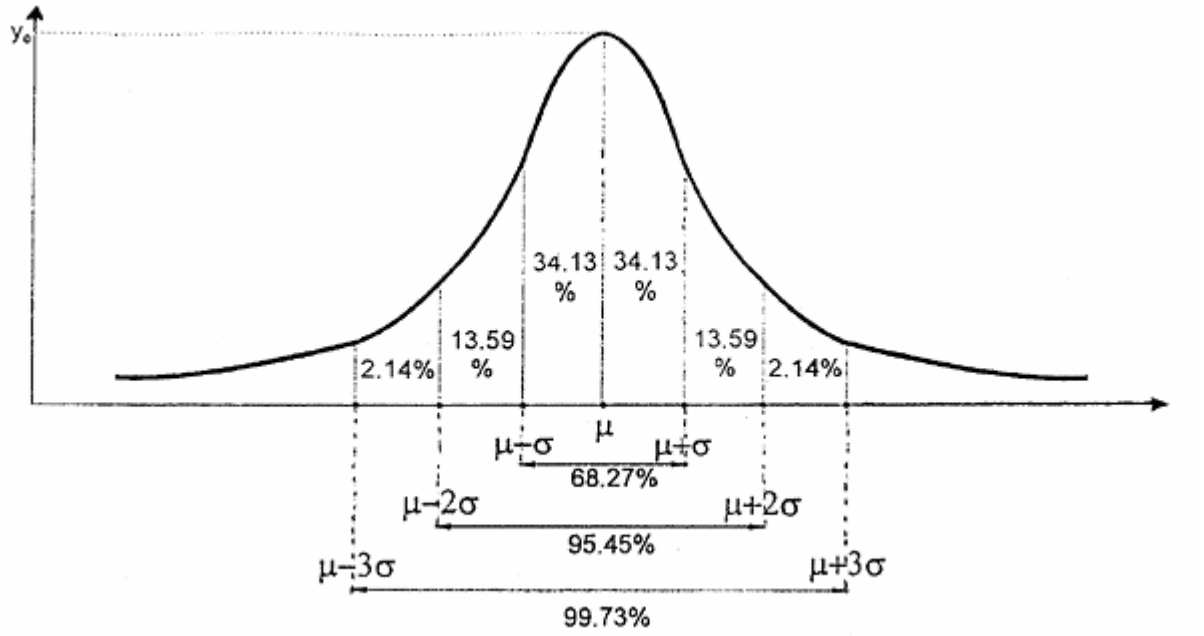


Рисунок 3.7. Нормальний розподіл [24].

Фільтр Гаусса модифікує вхідний сигнал згорткою з функцією Гауса [35].

Дискретний фільтр Гауса широко використовується для зменшення шуму. Цей фільтр має імпульсну відповідь:

$$h(n) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{1}{2}\left(\frac{\tau n}{\sigma}\right)^2}, \qquad (3.11)$$

де τ > 0 - деякий інтервал вибірки.

Цей фільтр підкреслює значення сигналу біля значення x(n), таким чином, він відрізняється від усереднювального фільтра, який вважає всі величини однаково цінними в околиці [x (n - N), ..., x (n + N)] навколо x (n), а також не дозволяє виникати новій структурі сигналу в міру збільшення масштабу згладжування [3].

3.3.4. Усереднення

Усереднення - це спосіб збільшення співвідношення сигналу до шуму у тих випадках, коли частотний спектр шуму і сигнал перекриваються. У цих випадках звичайна фільтрація не допомагає, оскільки частота фільтра встановлена для відхилення шуму, відкидає також сигнал.

Усереднення застосовується лише до сигналів, що повторюються, коли є набір даних разом із точною інформацією про час, щоб відстежувати точний момент, коли сигнал починається або переступає деякий поріг. Усі ці записи (свіпи, sweeps) підсумовуються, потім діляться на загальну кількість записів (свіпів) (N), щоб утворити середнє значення.

До остаточного поділу амплітуда сигналу події в загальній сумі записів збільшиться у N разів. Оскільки шум у кожному свіпі випадковий і не співвідноситься із шумом у будь-якому іншому записі, амплітуда шуму в накопиченому сигналі матиме збільшення лише у √N разів. Після поділу сигнал матиме величину одиниці, тоді як шум матиме величину 1/√N. Таким чином, співвідношення сигнал-шум збільшується на √N [14].

Для усереднення в програмі використана функція np.avarage з бібліотеки NumPy. Ця функція розраховує середнє зважене, точніше середнє арифметичне зважене.

Для дійсних чисел $x_1,\ldots,x_n$ з ваговими коєфіцієнтами $w_1,\ldots,w_n$ визначається як

$$\bar{x} = \frac{w_1 x_1 + w_2 x_2 + \cdots + w_n x_n}{w_1 + w_2 + \cdots + w_n}. \qquad (3.12)$$

Коли всі вагові коефіцієнти рівні між собою, $w_1 = w_2 = \ldots = w_n$, середнє арифметичне зважене буде дорівнювати середньому арифметичному [19].

Для автоматизації обробки даних був визначений взаємозвязок між параметрами і станом клітини, фізичний зміст цих параметрів. Після окреслення та аналізу необхідних розрахункових можна перейти до автоматизації обробки даних експерименту whole cell.

3.5. Вибір мови програмування та бібліотек

Python — інтерпретована об'єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією. Розроблена в 1990 році Гвідо ван Россумом. Python підтримує модулі та пакети модулів, що сприяє модульності та повторному використанню коду. Інтерпретатор Python та стандартні бібліотеки доступні як у скомпільованій, так і у вихідній формі на всіх основних платформах. В мові програмування Python підтримується кілька парадигм програмування, зокрема: об'єктно-орієнтована, процедурна, функціональна та аспектно-орієнтована [46].

Python має ефективні структури даних високого рівня та простий, але ефективний підхід до об'єктно-орієнтованого програмування. Синтаксис Пітона, динамічна обробка типів, а також те, що це інтерпретована мова, робить її зручною для написання скриптів та швидкої розробки прикладних програм у багатьох галузях на більшості платформ [4].

На Python можна написати вебдодатки, ігри, скрипти по автоматизації, комплексні системи розрахунку, системи управління життєзабезпеченням і багато іншого [26].

За рахунок простоти коду, подальший супровід програм, написаних на Python, легше (в порівнянні з Java, C++, Perl, PHP та ін.). Простий синтаксис дозволяє легко читати чужий код, розбиратися в давно написаному власному коді. Завдяки цьому при необхідності можна додавати нові параметри розрахунків та з легкістю змінювати попередні.

Величезна кількість бібліотек дозволяє науковцям працювати з таблицями Excel, обробляти експериментальні дані та робити розрахунки на рівні MATLAB.

Інтерпретатор мови Python і багата стандартна бібліотека можуть бути отримані з сайту Python, і можуть вільно розповсюджуватися. Цей сайт має дистрибутиви та посилання на численні модулі, програми, утиліти та додаткову документацію.

Інтерпретатор може бути легко розширений функціями та типами даних, розробленими на C чи C++. Python також зручний як мова розширення для прикладних програм, що потребують подальшого налагодження [4].

Python реалізований під усіма поширеними операційними системами і на безлічі архітектур Windows, Linux, MacOS, навіть на міні-комп'ютерах Arduino.

Мова Python дає великі можливості, оскільки відкриває доступ до основних необхідних науковцям формул і алгоритмів. Розглянемо обрані для розрахункового комплексу бібліотеки і обгрунтуємо доцільність вибору на декількох прикладах їх роботи.

Сторонні бібліотеки:

- pyABF - пакет Python, який забезпечує зчитування електрофізіологічних даних з файлів Axon Binary Format (ABF) версій 1 і 2,
- NumPy – бібліотека, що забезпечує підтримку багатовимірних масивів і містить безліч функцій і операторів для роботи з ними,
- Matplotlib – бібліотека для візуалізації даних двомірною і трьохвимірною графікою,
- PyQt5 – модулі графічного фреймворку Qt для мови Pyton.

Стандартні бібліотеки:

- sys – забезпечує функції і константи для взаємодії з інтерпретатором Pyton,

- itertools – модуль для створення циклів ітерації.

3.6. Розрахунки параметрів з допомогою бібліотек для Python

3.6.1. Тест мембрани. Метод аналізу

У розробленій програмі реалізовано алгоритм тестування мембрани, згідно з наведеними вище розрахунковими формулами, на такі параметри, як струм витоку (утримання, holding) I_H, опір мембрани R_M, опір доступу R_A та ємність мембрани C_M. Тестування мембрани відбувається на початку кожного треку, коли нейрон знаходиться в конфігурації цілої клітини (whole-cell) в режимі фіксації напруги (voltage-clamp). Реєструється відповідь на зміну напруги, на якій утримується клітина. Зміна напруги відбувається у вигляді квадратного імпульсу -10мВ, тривалістю 20мс. Паралельно з цим вимірюється відповідна зміна струму, що протікає крізь мембрану клітини (рис. 3.8). За характеристиками трансмембранного струму будуть вираховуватися згадані вище параметри. Наступна демонстрація алгоритму обчислення проводитиметься для реєстрації струму пірамідального нейрону, зображеного на лівому графіку рисунку 3.8 [39].

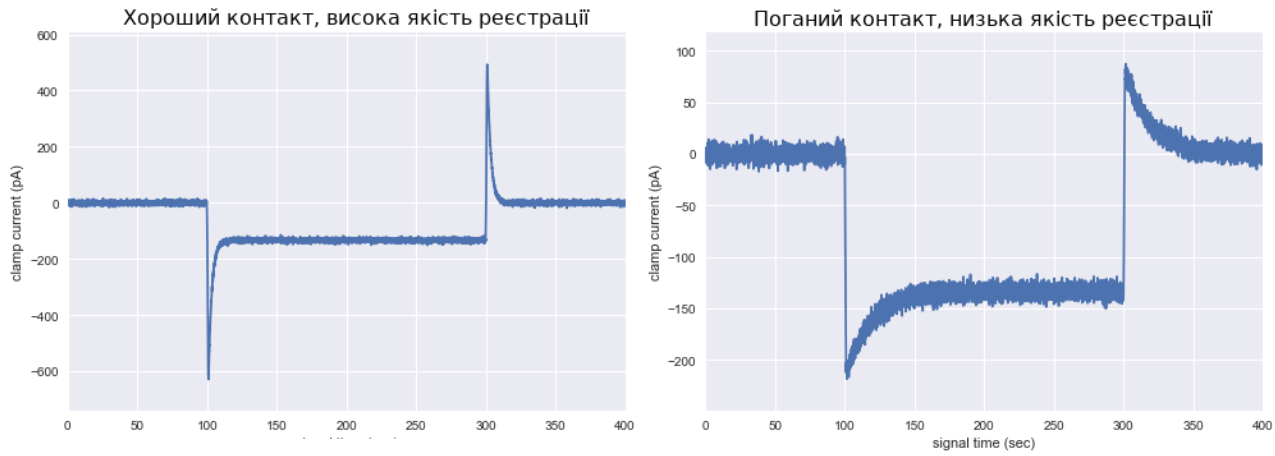


Рисунок 2.8. Відмінність ємнісного струму у випадку хорошого контакту піпетки з клітиною та у випадку поганого контакту з клітиною. На графіках зображена крива струму, що виникає у мембрані у відповідь на прикладення квадратного імпульсу напруги -10 мВ. По осі абсис відкладено час (мс), по осі ординат — силу струму (nA) [39].

3.6.2. Етапи обчислень у алгоритмі тестування мембрани

1) Розрахунок струму утримання I_H та опору мембрани R_M.

Розрахунок I_H і R_M опирається на усереднення та подальше обчислення різниці між усередненими струмами за проміжки часу до подачі тестуючого імпульсу напруги та під час тестового імпульсу, після виходу струму на плато (рис. 3.9).

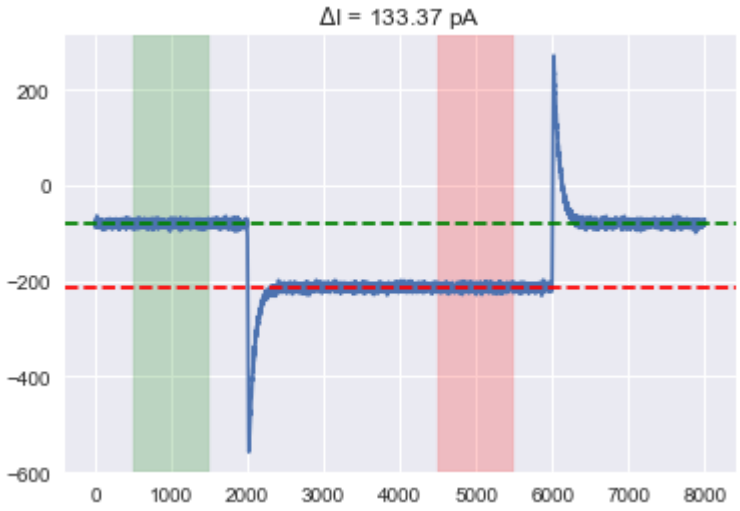


Рисунок 3.9. Виділені області — діапазони усереднення значень струму які порівнюються. По осі абсис відкладено час (мс), по осі ординат — силу струму (nA) [39].

Усереднення значень струму на обраних діапазонах:

```
I1=np.average(trace_full[500:1500]) #Виділена смуга зліва на рисунку 3.3.
I2=np.average(trace_full[4500:5500])#Виділена облать справа на рисунку 3.3.
```

I_H дорівнюватиме усередненому значенню струму в першій виділеній області:

```
Ih = I1*1e-12
```

Результат обчислення:

I_H= -79.956 pA.

R_M дорівнюватиме величині тестуючого імпульсу напруги поділеній на різницю між усередненими струмами на виділених двох часових діапазонах (3.8).

Визначення різниці між усередненмим силами струму на цих двох діапазонах:

```
dI = np.abs(I1-I2)*1e-12.
```

Результат обчислення:

ΔI = 133.37 pA.

Визначення R_M діленням величини тестуючого імпульсу напруги на ΔI:

```
dV = 10*1e-3
```

```
Rm = dV/dI
```
[39].

Результат обчислення:

R_M = 74.9 Mohm.

2) Розділення масиву на окремі секції.

Для подальшого розрахунку опору доступу (access resistance, R_A) та ємності мембрани C_M, необхідно розділити масив з даними результуючого струму на окремі ділянки у відповідності з початковою та кінцевою точкою прикладення тестуючого імпульсу напруги ΔV = 10мВ (рис. 3.10).

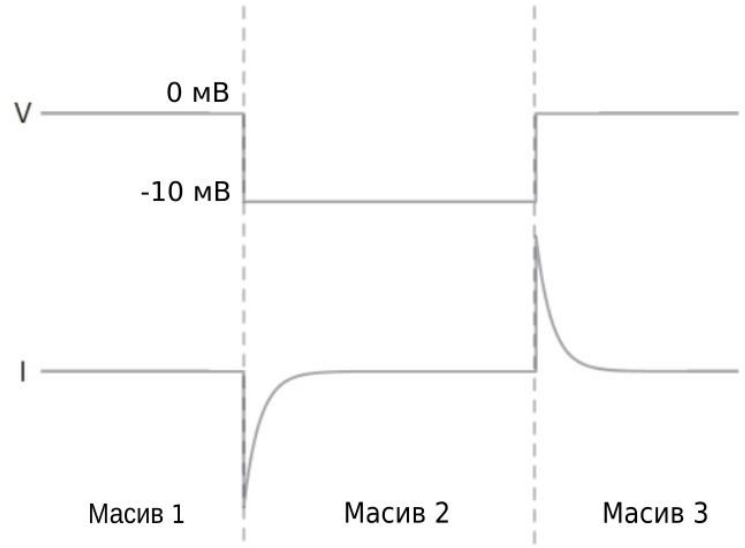


Рисунок 3.10.

Розбиття даних на окремі масиви задля подальшої обробки їх як оккремих відбувається за двома осями: trace_full — значення струму (вісь ординат), trace1, trace2, trace3 — перший, другий та третій масиви відповідно. Позиції 2000 та 6000 в масиві trace_full — початкова та кінцева точки прикладення тестуючого імпульсу напруги (рис. 3.11):

```
trace1=trace_full[:2000]
trace2=trace_full[2000:6000]
trace3=trace_full[6000:]
```
[46].

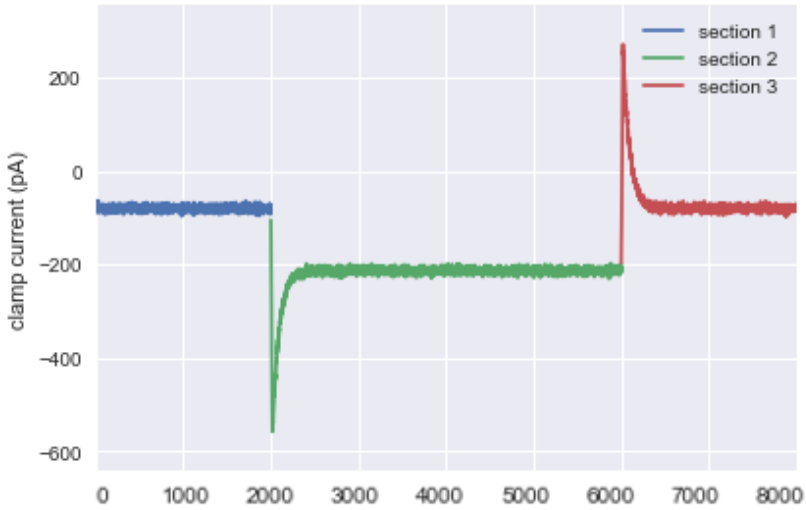


Рисунок 3.11. Розбиття масиву значень струму на окремі масиви. По осі абсис відкладено час (мс), по осі ординат — силу струму (нА) [39].

3) Розрахунок опору доступу (access resistance, R_A) за амплітудою ємнісного струму.

Як було зазначено у другому розділі, є два способи розрахунку опору доступу: за фактичною амплітудою ємнісного струму та за теоретичним розрахунком.

Рахунок опору доступу (access resistance, R_A) методом визначення амплітуди ємнісного струму відбувається за знаходженням мінімального значення струму у другому масиві (див. рис. 3.10), що дорівнює амплітуді транзієнтного (ємнісного) струму ΔI_A (рис. 3.11, крива між 2000 мс і 6000 мс), й подальшого ділення на нього величини тестуючого імпульсу напруги ΔV =

10 мВ. Обчислення відбувається за формулою (3.8).

Зміну ємнісного струму ΔI_A знаходимо відніманням середнього значення струму першого масиву від мінімального значення струму у другому масиві:

```
I1 = np.average(trace1)
I2 = np.min(trace2)
dIa = (I2-I1)*1e-12
```
[39].

Результат обчислення:

ΔI_A = - 560 - (-80) = -480 pA.

Знаходження опору доступу (access resistance, R_A) за законом Ома:

```
dV = 10*1e-3
Ra = dV/dIa
```
[39].

Результат обчислення:

R_A = 10mV/480pA = 20Mohm.

Оскільки даний метод обчислення опору доступу дає точний результат тільки у випадку, коли ведеться обробка нефільтрованих данних, в розробленій програмі для обробки даних цей метод був замінений теоретичним методом обчислення опору доступу з допомогою сталої часу, який буде розглянуто далі.

4) Розрахунок сталої часу мембрани τ.

Для подальшого розрахунку використовуються дані з третього масиву trace3 (див. рис. 3.10), а саме ділянки експоненційного спадання транзієнтного (ємнісного) струму.

Для обчислення постійної часу мембрани (τ) необхідно виділити лише ділянку експоненційного спадання ємнісного струму, знехтувавши при цьому ділянкою наростання цього струму, та ділянкою після досягнення струмом плато, рівень якого приймається за нульове значення. Для виокремлення ділянки експоненційного спадання треба спершу визначити у масиві позиції, що обмежують дану ділянку. Перша, з якої починається експоненційне

спадання - позиція максимального значення струму у заданому масиві. Її значення присвоюється змінній peakI. Друга позиція позначає завершення експоненційного спадання. Позиція позначає точку, на якій значення струму досягає нульового значення (вихід на плато, який надалі приймається за нульове). Її значення присвоюється змінній zeroI.

```
originalCurve = trace3
peakI=np.where(originalCurve==np.max(originalCurve)
)[0][0] # Повертає позицію максимального значення
струму.
zeroI=np.where(originalCurve[peakI:]<=0)[0][0]+peak
I # Повертає позицію де закінчується спадаюча крива.
```

Наступним кроком дані значень струму, обмежені цими двома позиціями, переносяться у окремий масив fitThis для подальшого знаходження сталої часу мембрани (τ).

```
fitThis=originalCurve[peakI:zeroI]
```

Також створюється масив значень часу у секундах, fitThisXs, що відповідають значеням струму у масиві fitThis. Число 20000 означає частоту дискретизації підсилювача, і являє собою кількість зареєстрованих значень струму на секунду.

```
fitThisXs=np.arange(len(fitThis))/20000.
```

Після цих дій отримаємо масив значень що на рисунку 3.6 знаходиться між точками.

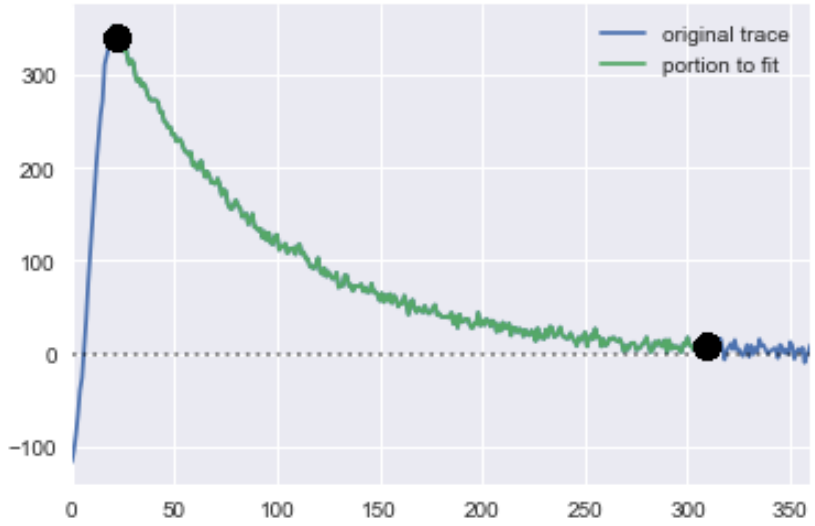


Рисунок 3.12. Ємнісний струм, що характеризує ємність мембрани клітини (C_M). Середня ділянка — діапазон обчислння сталої часу τ. По осі абсис відкладено час (мс), по осі ординат — силу струму (нА) [39].

Після того, як ділянка експоненцйного спадання струму відокремлена в масив fitThis, її константа часу (τ) може бути знайдена за допомогою апроксимації до неї експоненти. З формули (3.4), приймаючи час t за x, експонента описується лише значенням τ, до підбору якого зводиться задача.

Наведений нижче код реалізує алгоритм послідовної апроксимації заданої кривої експонентою, що описується формулою (3.4).

Спершу оголошується функція monoExp1d, що повертає експоненту від переданих у якості аргументів масиву значень *x* та константу τ, та описує графік експоненти, якою апроксимуються дані. Змінні Xs та tau — відповідно *x* і τ:

```
def monoExp1d(Xs, tau):
    return np.exp(-Xs/tau)
```

Наступною оголошується функція, що розраховує значення похибки (змінна err), яка визначається як сума різниць між значенням струму та відповідної точки апроксимованої експоненти для кожного зі значень *x*. Функція повертає значення err. На вході функція як аргументи приймає масиви значень *x* та *y* через змінні Xs та data відповідно, а також константу τ.

```
    def monoExp1dErr(Xs, data, tau):
        fitted=monoExp1d(Xs,tau)  # Отримуємо масив значень експоненти з заданою сталою часу.
        err=np.sum(fitted-data) # Підсумовує різниці між експонентою і експериментальною кривою.
    return err
```

Наступна, остання оголошена функція, виконує підбір значень tau, додаючи до нього, чи віднімаючи від нього в кожній наступній ітерації значення step, що за замовчуванням дорівнює 0,5. В кожній ітерації викликається функція monoExp1dErr, що повертає значення похибки при даній tau. Якщо отриманне значення похибки менше за нуль, це означає, що експонента, що апроксимується до даних, знаходиться занадто низько і tau збільшується на значення step. Навпаки, якщо отриманне значення похибки більше за нуль, це означає, що експонента, що апроксимується до даних, знаходиться занадто високо і tau зменшується на значення step. Нарешті, якщо в кожній наступній ітерації знак похибки змінюється на протилежний, це означає, що за даного значення кроку, коефіцієнт tau підібрано максимально точно і для подальшого зменшення похибки необхідно зменшити крок підбору у два рази:

```
    def bestTau(Xs, data, tau=.1, step=.5):
        errs=[np.inf] # Оголошення списку похибок.
        normed=data/data[0] # Нормалізація вхідних даних (зведення до значень в межах від 0 до 1).
        while np.abs(errs[-1])<0.01 or len(errs)<100: # Підбір значення tau допоки похибка не стане меншою 0,01 чи цикл не виконає 100 ітерацій.
            err = monoExp1dErr(Xs,normed,tau) # Знаходження поточної похибки та додавання її до списку.
            errs.append(err)
```

```
            if (errs[-1]>0 and errs[-2]<0) or (errs[-
1]<0 and errs[-2]>0): # Зменшення кроку підбору в два
рази якщо дві останні похибки мали протилежні значення.
                step/=2
            if errs[-1]<0: # Зміна tau на значення step
в залежності від знаку останньої похибки.
                tau+=step
            elif errs[-1]>0:
                tau-=step
        return tau
    tau=bestTau(fitThisXs,fitThis)
```

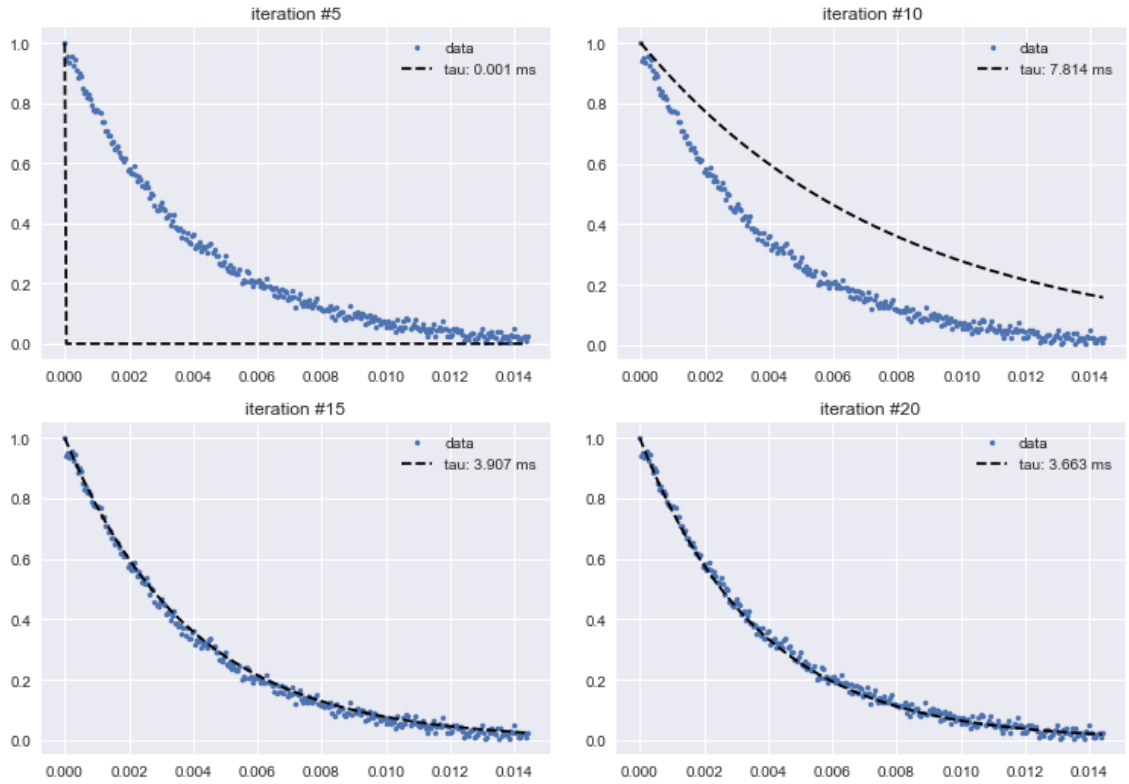


Рисунок 3.13. Поетапна апроксимація експоненти, представленої пунктирною лінією, до значень спадаючого експоненційного струму, представленого точками, для п'ятої, десятої, п'ятнадцятої та двадцятої ітерації з послідовним зменшенням похибки. По осі абсис відкладено час (с), по осі ординат — силу струму (нормалізована до одиниці шкала) [39].

На вході функція отримує значення масивів fitThis зі значеннями струму та fitThisXs з відповідними значенням часу. Функція повертає значення τ в секундах, для апроксимованої експоненти [39].

Тепер, після того, як знайдена постійна часу мембрани τ, можна легко знайти значення опору доступу (access resistance, R_A) та ємності мембрани (membrane capacitance, C_M).

5) Розрахунок опору доступу (access resistance, R_A) та ємності мембрани (membrane capacitance, C_M) на основі апроксимованої експоненти через постійну часу мембрани τ.

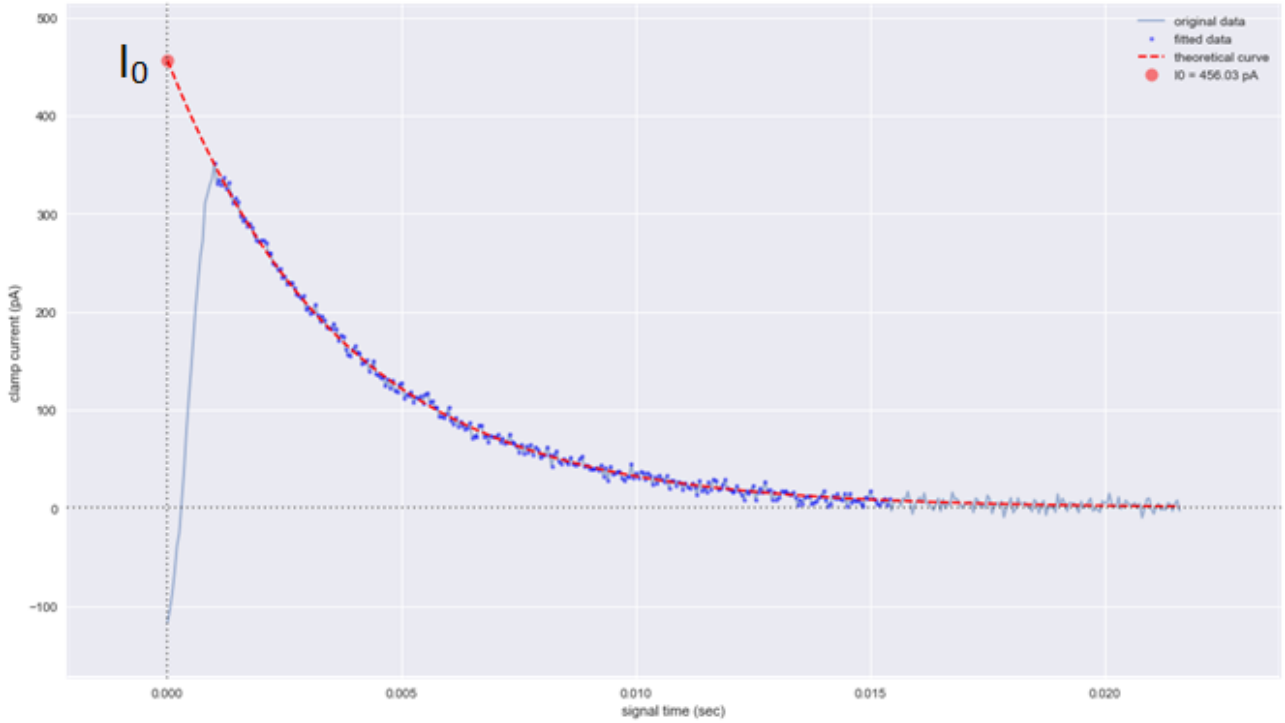


Рисунок 3.14. Точки на графіку — значення струму на ділянці експоненційного спадання, до яких проводиться наближення експоненти. Пунктирна крива — експонента, підібрана до експоненційного спадання струму. I_0 — початкове значення ємнісного (транзієнтного) струму, вираховане за допомогою наближеної експоненти. По осі абсис відкладено час (с), по осі ординат — силу струму (нА) [39].

Знайдена константа часу τ, що описує апроксимовану експоненту яку було підібрано до експоненційно спадаючого струму, дозволяє теоретичним

шляхом визначити I_0 – амплітуду ємнісного струму, знайдену аналітично, - підставивши у формулу (3.4), а в якості значення t - час в який було прикладено тестуючий імпульс напруги.

```
fittedXs=(np.arange(int(len(fitThis)*1.5))-peakI)/20000 # Створення масиву часу для побудови наближеної експоненційної кривої.
fittedCurve=monoExp1d(fittedXs,tau)*fitThis[0] # Використання tau для побудови теоритичної експоненти до точки I0.
I0=fittedCurve[0]*1e-12 # Пошук точки I0 на апроксимованій кривій.
```

Знаючи теоретично розраховане значення амплітуди ємнісного струму, можна визначити опір доступу за формулою (3.8):

```
Ra=dV/I0 # Розрахунок Ra.
```

Останнім кроком, з відомого опору доступу, за формулою (3.7) стає можливо розрахувати ємність мембрани:

```
Cm=tau/Ra # Розрахунок Cm
```
[39].

Всі параметри, а саме опір доступу (R_A), ємність мембрани (C_M), струм утримання (I_H) та опір мембрани (R_M) що були розписані вище, розраховуються для кожного треку і виводяться у вигляді графіку для кожгого файлу, що аналізується (див. Розділ 4).

3.7. Контроль інформації що вводиться і діагностика помилок

Помилки, виявлені під час виконання, називаються винятками. Винятки бувають різних типів [34]. Для обробки винятків використано конструкцію try - except [20]. В розробленій програмі користувач вносить назву, або перелік назв файлів для обробки. В разі введення помилкових даних, помилка буде перехоплена і користувачу буде повідомлено про помилкове введення.

Блок try дозволяє перевірити блок коду на наявність помилок [47]. У

блоці try виконується інструкція, яка може породити помилку, а в блоці except вона перехоплюється. При цьому перехоплюється як саме виключення, так і його нащадки [20].

Якщо під час виконання блоку try відбувається виняток, решта цього блоку пропускається. У випадку, коли тип винятку відповідає вказаному в except, виконуються інструкції блоку except, а потім виконання продовжується після оператора try [34]. Також можлива інструкція except без аргументів, яка перехоплює будь-які винятки [20].

Якщо трапляється виняток, який не зазначено в except, він є необробленим винятком і виконання зупиняється. Виконання try операцій закінчується лише коли виняток не відбувається.

Оператор try-except має необов'язковий блок, який, за наявності, повинен бути виконаний, якщо блок try не викликав винятку. Використання else краще, ніж додавання додаткового коду до try, оскільки це дозволяє уникнути випадку виникнення непередбачуваного винятку [34].

Блок finally дозволяє виконати код, незалежно від результату блоку try [47]. Він виконує інструкції в будь-якому випадку, чи було виключення, чи ні і може бути застосований, коли потрібно неодмінно щось зробити, наприклад, закрити файл [20].

Висновки до третього розділу

Для розробки програми було використано всі необхідні для розрахунків можливості бібліотек для мови Python, таких як pyABF, sciPY, matplotlib, PyQt5 та numpy. Було уважно перевірено правильність розрахунків програмою, після чого було змінено спосіб розрахунку струму для визначення опору доступу. Якість розрахунку інших параметрів була задовільна.

Для зручності і можливості використання програми іншими науковцями, інтерфейс Qt Designer було інтергровано в програму з допомогою фреймворку PyQt5, в результаті з'явилась необхідність додати більше режимів

і розрахункових параметрів, що буде детальніше розглянуто в наступному розділі.

Загалом, усі необхідні потреби в автоматизації обробки електрофізіологічних даних для заданої конфігурації експерименту програма задовольняє, тому на цьому етапі немає потреби вдосконалювати цей обробний комплекс. Однак простота коду, завдяки великій кількості застосованих бібліотечних функцій і нескладному синтаксису обраної мови програмування Python, дозволяє в подальшому змінювати і вдосконалювати код будь-кому для своїх наукових потреб.

РОЗДІЛ 4. ВЗАЄМОДІЯ З СИСТЕМОЮ ОБРОБКИ ДАНИХ

Програма орієнтована на застосування студентами та аспірантами електрофізіологами чи біофізиками, що беруть участь в проектах, де ведуться дослідження за методикою patch-clamp.

4.1. Головне вікно

Головне вікно програми представляє собою набір елементів керування для задання параметрів візуалізації даних з файлів формату ABF. Також реалізовано тестування мембрани з подальшим виводом інформації у вигляді графіку. Оскільки при роботі з електрофізіологічними даними часто доводиться застосовувати однакові операції до великої кількості однотипних даних, в програмі реалізована пакетна обробка файлів за вказаними параметрами.

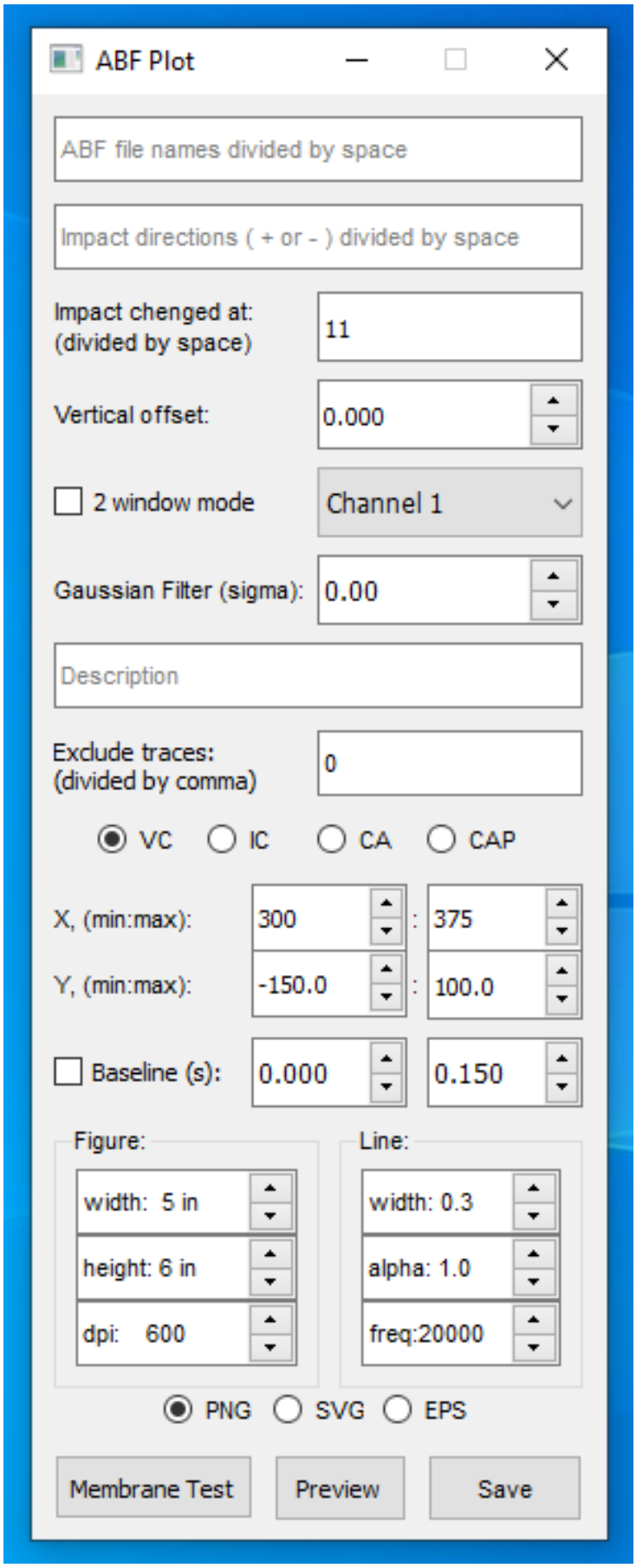


Рисунок 3.1. Головне вікно програми.

4.2. Задання файлу або списку файлів для обробки

Головний параметр, який необхідно задати для початку роботи з програмою - це ім’я файлу без розширення, або кількох файлів, необхідних для обробки, перелічені через пробіл. Для цього використовується перше верхнє поле у вікні програми (див. рис 4.1). Слід зазначити, що для правильної роботи програми необхідно, щоб імена файлів не містили пробілів (рис 4.2). Зазвичай це не є проблемою, адже файли з даними, що створюються під час електрофізіологічного досліду іменуються автоматично. Крім того, файли мають знаходитись в тій самій директорії, що й запущений екземпляр програми.

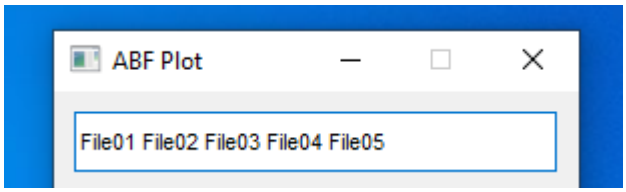


Рисунок 4.2. Приклад правильно заповненого списку файлів.

4.3. Тест мембрани

Для проведення тесту мембрани немає необхідності вказувати додаткові параметри, крім імен файлів, що підлягають обробці. Після того, як список файлів оформлений у список за зазначеними вище правилами, достатньо запустити обробку натиснувши кнопку Membrane test (див. рис 4.1).

Для кожного з вказаних у списку файлів програма будує по 4 графіки, по одному для кожного з визначених значень, а саме опір доступу R_A, ємність мембрани C_M, струм утримання I_H та опір мембрани R_M, що були розписані в розділі 3. Ці параметри розраховуються для кожного треку і виводяться у вигляді графіку для кожного файлу, що аналізується, з подальшою

можливістю їх збереження у векторних (SVG, EPS) чи растрових (PNG) форматах, або PDF. По осі ординат відкладається значення поточного параметру для кожного з треків, а по осі абсцис — час коли було здійснено замір впродовж однієї експериментальної реєстрації (в межах одного файлу).

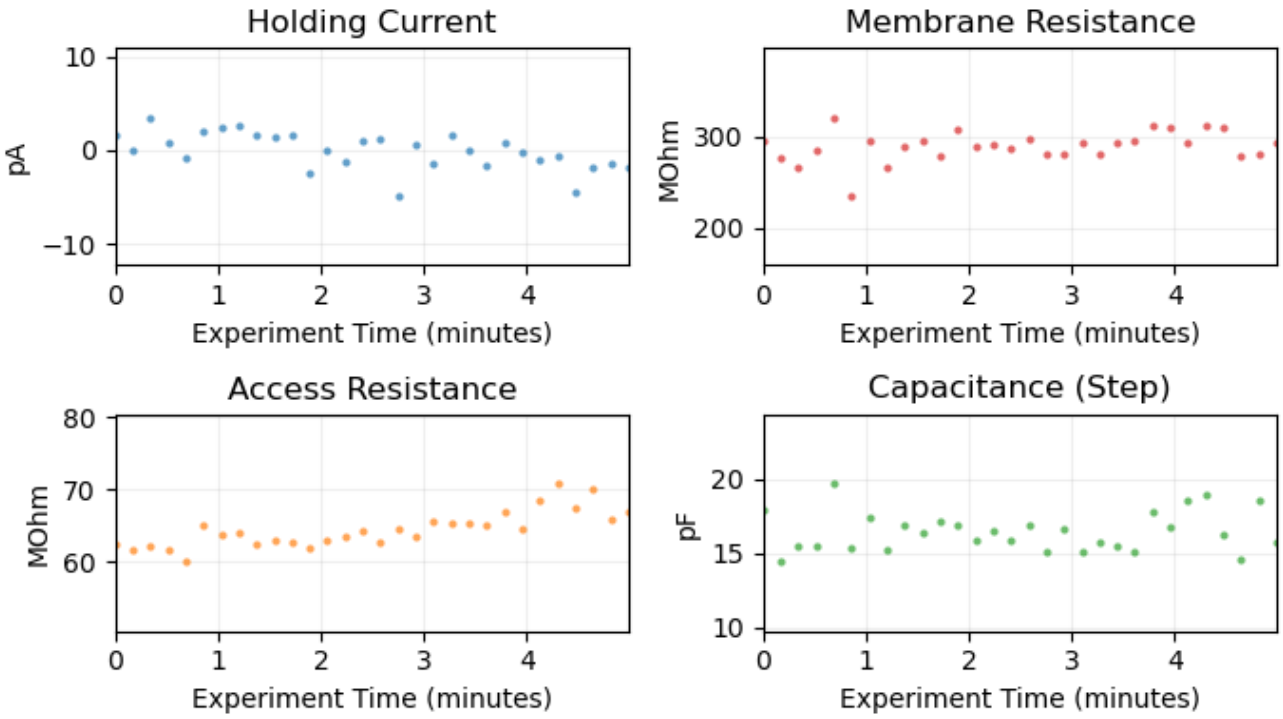


Рисунок 4.3. Графічне представлення обчислених програмою параметрів клітини. По осі абсцис відкладений час, протягом якого вимірювались параметри клітини.

Кожна з точок на графіках тесту мембрани це один трек (трейс) з файлу записів. Якщо серед записів зявиться трек з будь-якими параметрами, що не відповідають вимогам експериментальної вибірки, його можна буде виключити вказавши в полі Exclude traces, вказуючи номери треків через кому.

Рисунок 4.4. Поле для виключення треків.

4.4. Візуалізація даних електрофізіологічно досліду

Основна функція програми - візуалізація та збереження даних отриманих у процесі електрофізіологічних дослідів. Для графічної візуалізації даних після вводу імен файлів, що підлягають обробці, достатньо натиснути кнопку Preview. Виведений у новому вікні графік можна вільно масштабувати чи зберегти (див. Рис 4.4). Слід зазначити, що якщо не змінювати інших параметрів у головному вікні, при побудові графіків використовуватимуться параметри за замовчуванням.

Параметри, які можна змінювати наступні:

- Impact changed at (рис. 4.5) - поле, значення в якому вказують на трек у поточному файлі, починаючи з якого знак стимуляції клітини змінювався на протилежний. Вказаний трек і всі наступні після нього для відповідного файлу будуть зафарбовані іншим кольором (рис. 4.6). Поле дозволяє ввести параметри для кількох файлів, розділені пробілом, у порядку відповідно до введеного раніше списку файлів. Кількість вказаних параметрів та їх порядок має відповідати кількості та порядку вказаних для обробки файлів (див. рис. 4.5). Значення за замовчуванням – 0.

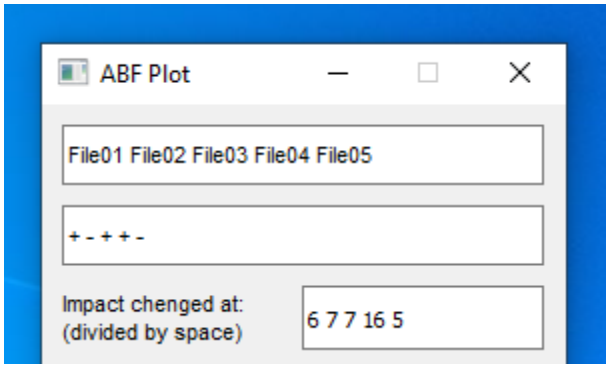


Рисунок 4.5. Приклад заповнення списку файлів та параметрів їх обробки: кількість вказаних параметрів та їх порядок має відповідати кількості та порядку вказаних для обробки файлів.

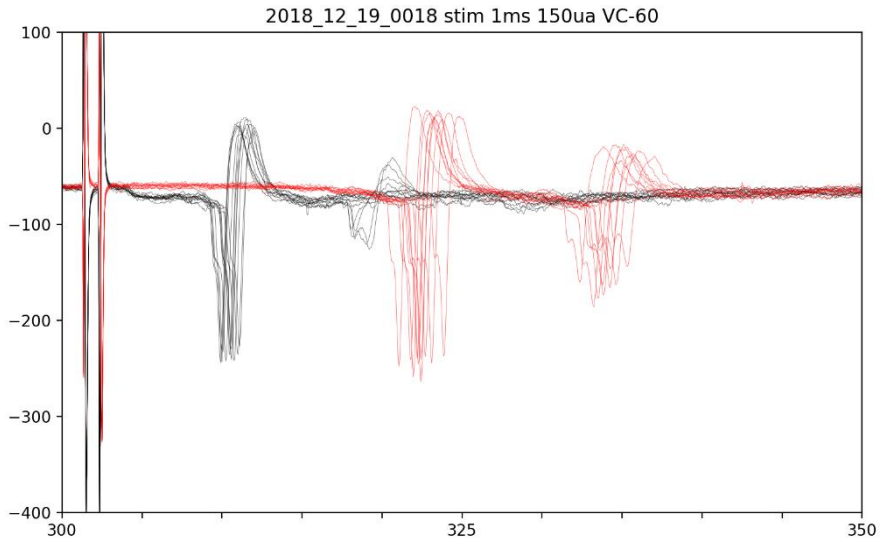


Рисунок 4.6. Приклад отриманого зображення з різним забарвленням треків починаючи з номеру вказаного.

- Impact directions (див. рис. 4.1) - полярність стимуляції. Параметр вказує на те, якого кольору будуть зафарбовуватися треки до зміни полярності стимуляції. Тобто, якщо вказано “+”, то треки, що йдуть до зміни полярності стимуляції (параметр Impact changed at) будуть зафарбовані чорним кольором, а решта червоним. При зміні цього параметра на “-” забарвлення буде протилежним. Як і у попередньому випадку, поле дозволяє ввести параметри для кількох файлів, розділені пробілом, у порядку відповідно до введеного раніше списку файлів. Кількість вказаних параметрів та їх порядок має відповідати кількості та порядку вказаних для обробки файлів. Значення за замовчуванням - “+”
- Vertical Offset - вертикальне зміщення кожного наступного треку відносно попереднього.

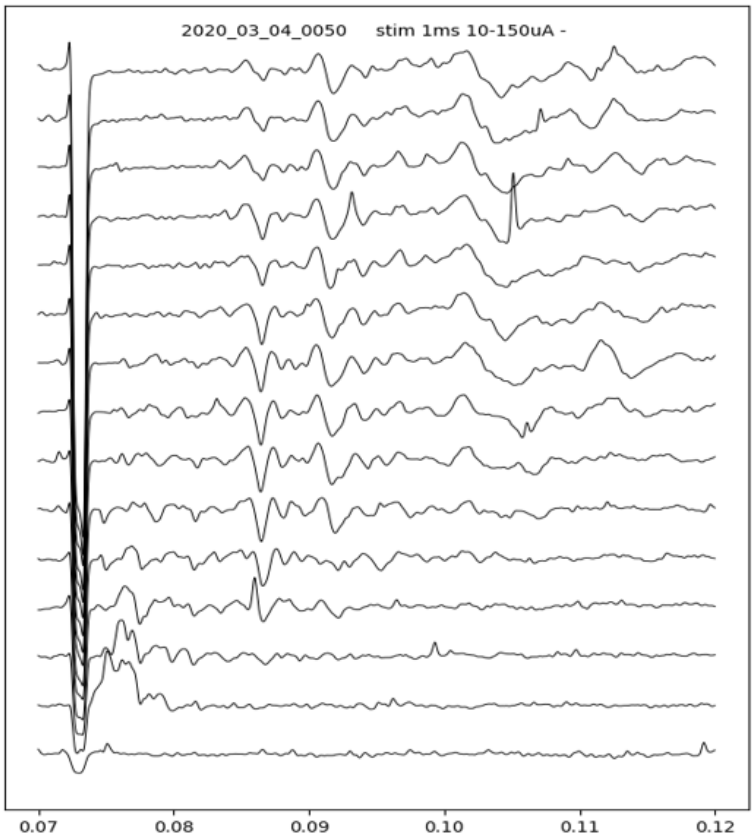


Рисунок 4.6. Приклад створеного графіку з вертикальним зміщенням.

- Channel - випадаючий список вибору каналу, з якого робився запис. Оскільки підсилювач може робити записи зразу з двох клітин в один файл, виникає необхідність переключатися між ними. Можливі варіанти нульового та першого каналу, а також обох, якщо робився двохканальний запис експерименту.

.

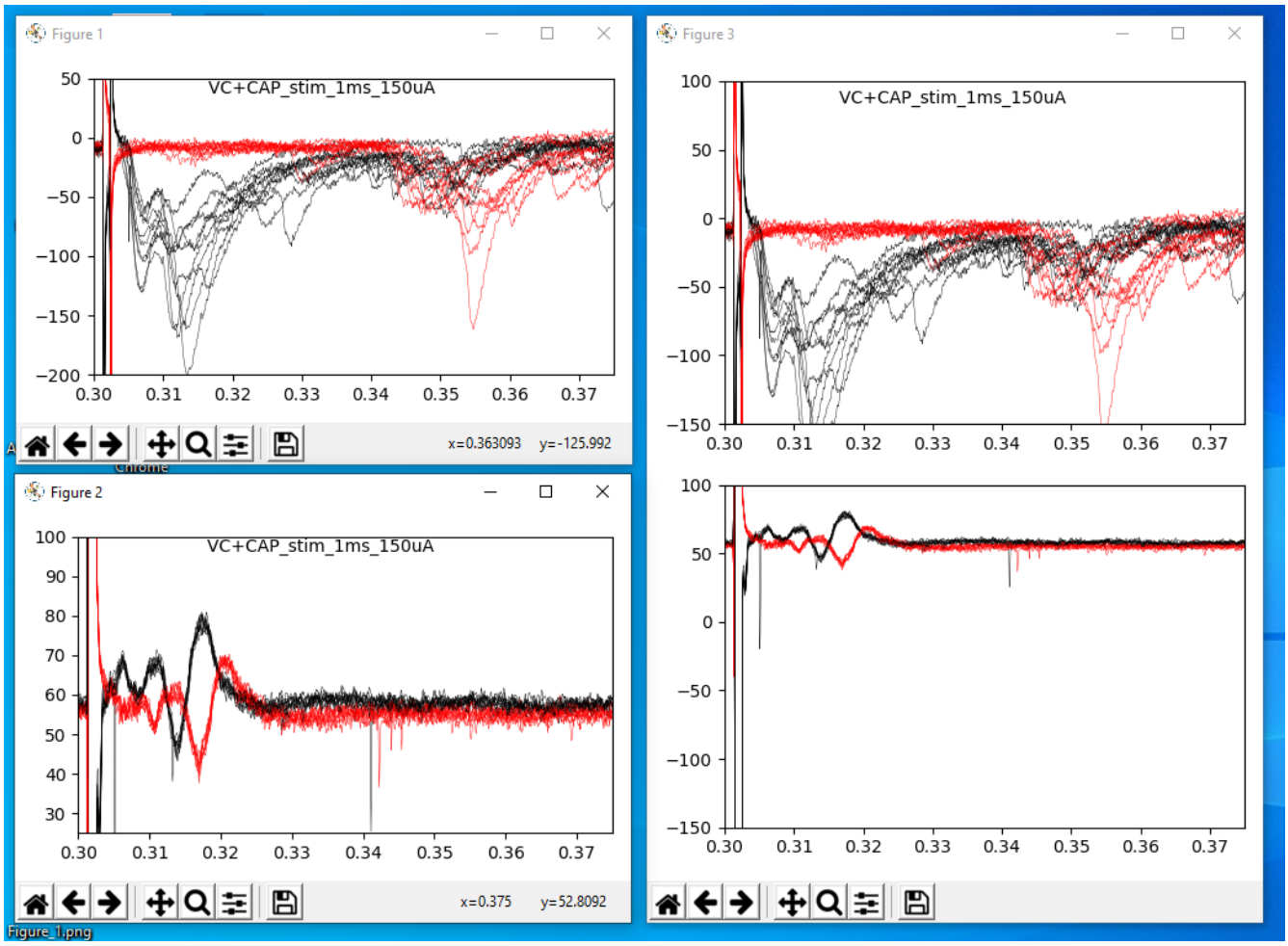


Рисунок 4.7. Приклад побудови графіку з нульового каналу (ліве верхнє вікно), першого каналу (ліве нижнє вікно), та з обох каналів одночасно (праве вікно).

- Gaussian filter - коефіцієнт фільтрації (сигма) для фільтра Гауса.

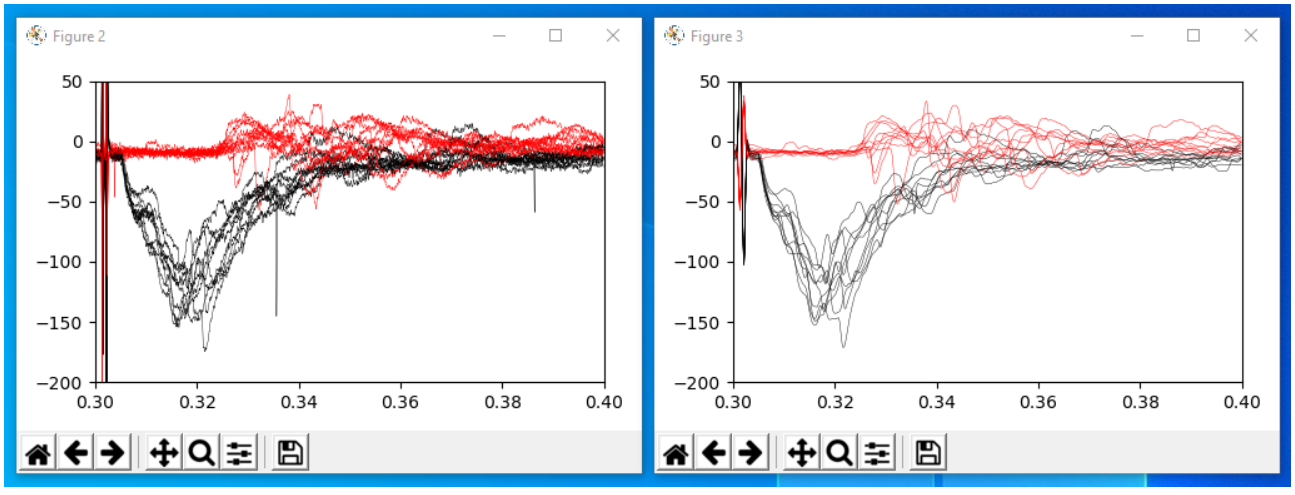


Рисунок 4.8. Порівняння результату без фільтрації (зліва) та з

фільтрацією σ=0.15 (справа).

- Межі побудови графіку - значення початкових і кінцевих координат (по вісі абсцис в мс, по вісі ординат пА або мВ), якими обмежується інформація, що цікавить. Цей параметр корисно використовувати, коли треба створити зображень для великої кількості файлів. Також наявні 4 перемикачі для встановлення найбільш розповсюджених конфігурацій експерименту з розповсюдженими параметрів масштабування.

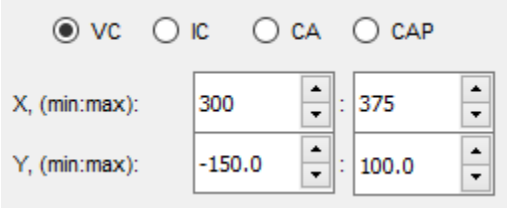


Рисунок 4.9. Налаштування меж побудови графіку та перемикачі, що автоматично виставляють найбільш росповсюджені параметри масштабування.

- Baseline - функція вирівнювання треків відносно нуля. В числових полях вказується межі проміжку треку (в секундах), що має вважатися за нуль. Цей параметр актуально використовувати коли треки мають незаплановане вертикальне зміщення між собою, що може бути наслідком поганого контакту піпетки з клітиною.

Рисунок 4.10.

- Параметри побудови зображення - налаштування ширини й висоти кінцевого зображення, його роздільної здатності (DPI - точок на дюйм), товщини та прозорість ліній треків, а також частота дискретизації (кількість точок по осі абсцис на секунду).

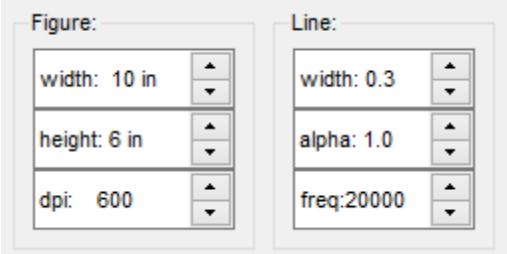


Рисунок 4.11. Налаштування параметрів кінцевого зображення.

4.5. Збереження результатів, формат вихідних даних

Програма дозволяє зберігати отриману графічну інформацію у файли. Це можливо зробити як в кожному окремому вікні, позиціонуючи та масштабуючи графік вручну, так і в автоматичному режимі, вказавши параметри масштабування для всього списку файлів. В другому випадку графіки збережуться у тій самій директорії, що й вихідні файли з даними, під тими самими іменами. Для збереження, після введення всіх параметрів, достатньо обрати необхідний формат та натиснути кнопку Save (див. рис. 4.1).

Програма на виході формує файл з зображенням графіків струму (voltage-clamp) [51] або напруги (current-clamp) [52] в форматах PNG, EPS, або SVG.

PNG (Portable Network Graphics) — растровий формат збереження графічної інформації, що використовує стиснення без втрат. Він використовує відкритий, не патентований алгоритм стиснення Deflate (алгоритм стиснення без втрат, що використовує комбінацію алгоритмів LZ77 і Хаффмана) [30], вільні реалізації якого доступні в інтернеті. PNG був створений для заміни формату GIF (8-бітний растровий графічний формат, що використовує до 256 чітких кольорів із 24-бітного діапазону RGB) [36] графічним форматом, який не потребує ліцензії для використання, отже може застосовуватись для відкритого ПЗ. Хоч PNG характеризується сильнішим рівнем стиснення для файлів з більшою кількістю кольорів ніж GIF, але різниця становить близько 5-25%, чого недостатньо для абсолютної переваги формату [44].

EPS (Encapsulated PostScript) — формат файлів, розроблений компанією Adobe Systems, закодований в машинні коди графічний файл який описано мовою програмування PostScript (мова програмування та розмітки сторінок, в основному застосовується у видавничих системах) [45]. Використовується переважно для друку, тому його доцільно використовувати для розміщення графіків на плакатах до наукових конференцій і доповідей. Містить як векторну інформацію так і растрову [33].

SVG (Scalable Vector Graphics) - мова розмітки масштабованої векторної графіки, створена Консорціумом Всесвітньої павутини (W3C), входить в підмножину розширюваної мови розмітки XML , призначеної для опису двовимірної векторної і змішаної векторно-растрової графіки в форматі XML. Підтримує як нерухому, так і анімовану інтерактивну графіку. Не підтримує опис тривимірних об'єктів. Це відкритий стандарт, який є рекомендацією консорціуму W3C - організації, яка розробила такі стандарти, як HTML і XHTML . В основу SVG лягли мови розмітки VML і PGML . Розробляється з 1999 року. На сьогодні актуальною є версія 1.1 2011 року [10].

4.6. Перехоплення помилок

В програмі реалізовано перехоплення помилок користувацького вводу, а також відсутності вказаного користувачем файлу (рис. 4.12). При введені помилкових даних програма не завершує роботу з наступною втратою всіх введених параметрів, а виводить повідомлення про необхідність перевірити їх правильність. Це реалізовано з допомогою конструкції try-exept, описаної в розділі 3.7.

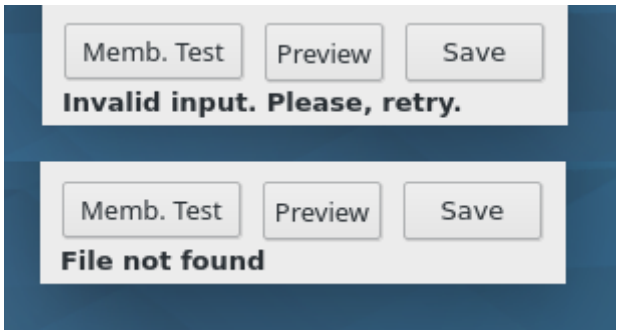


Рисунок 4.12.Помилка введення параметрів (угорі) та введення імені файлу/файлів (знизу).

Висновки до четвертого розділу

Дана програма допомагає пакетній обробці електрофізіологічних даних, отриманих в роботі методикою patch-clamp на обладнанні Axon і підходить для студентів та аспірантів, задіяних у дослідженнях в цій області. Програма розроблена з ціллю пришвидшити найбільш тривалу стадію дослідницької роботи - обробку отриманих даних.

Завдання, що переслідувались при розробці:

1) Побудова графіків.

Програма працює даними реєстрації струму (voltage-clamp) або напруги (current-clamp) на мембрані клітини.

Ці дані вона виводить у вигляді графіків залежності струму/напруги (по осі ординат) у пікоамперах/мілівольтах від часу (по осі абсцис) у секундах. Графіки можна будувати по одному, або декілька за раз, накладаючи один на одний графіки ряду треків (trace), чи те саме зі зміщенням.

Для пасивних параметрів клітини, таких як ємність мембрани (membrane capacitance) в пікофарадах, її опір (membrane resistance) в мегаомах і параметрів контакту мембрани клітини з піпеткою – струму витоку (leak current), пА, і опору доступу (access resistance), МОм – будує графік зміни цих параметрів в кожному з записаних треків (trace).

2) Усереднення.

З допомогою засобів бібліотеки numpy виконується усереднення

значень декількох треків (трейсів), в результаті чого можна отримати один графік зміни струму/напруги.

3) Виключення високочастотного шуму.

Для виключення високочастотного шуму використовується фільтр Гауса. Він модифікує вхідний сигнал наближаючи його до функції Гауса; це також відомо як перетворення Вейєрштраса [35].

4) Експорт у векторне зображення

Побудовані графіки зберігає у форматах PNG, EPS, або SVG.

5) Пакетна обробка.

Замість одного файлу, для обробки даних того самого протоколу, можна вказати декілька через пробіл. Пакетна обробка виконується з заданими одноразово параметрами для кожного з вказаних у файловому полі (ABF File Name) файлу даних.

В розділі було представлено результати роботи програми для кожної з її функцій. Усі параметри обробки наявні в графічному інтерфейсі, який є досить простий і лаконічний, завдяки чому кожен користувач може без підготовки працювати з програмою.

ВИСНОВКИ

Завданням проекту була розробка відкритої програмної системи для швидкої пакетної обробки електрофізіологічних даних науковцями, студентами та аспірантами, задіяними в цій галузі. Вузькоспецифічні задачі, які часто необхідні для цієї області, не завжди є реалізованими у готових ппрограмних продуктах, таких як, наприклад, Clampfit чи Origin. Крім того важливо обробляти дані за однаковим шаблоном в уніфікованому вигляді, а пакетна обробка даних вимагає знання відповідної мови програмування макросів, яка відрізняється у кожній з них.

Більшість часу наукової роботи займає обробка даних, тому проста в користуванні система зі всіма необхідними в даних дослідженнях задачами зекономить цей час. Оскільки програмний код є відкритим, це дає можливість іншим користувачам внести свої корективи в разі зміни протоколу експерименту, появи додаткових вимог до обробки, або будь-яких нових задач.

Система відповідає висунутим до неї технічним вимогам: дозволяє проводити пакетну обробку великої кількості файлів, в програмі реалізоване перехоплення помилок, що виникають внаслідок введення неправильних параметрів чи невідповідних вхідних даних. Це дозволяє уникнути аварійного завершення роботи програми з втратою введених користувачем параметрів обробки. В програмі реалізований вибір набору налаштувань за замовчуванням, що відповідають найросповсюдженішим потребам користувачів. Також програма дозволяє робити попередній перегляд та ручну корекцію візуалізованих графічних даних. Крім того, вона є кросплатформеною та може процювати у всіх актуальнийх на даний час операційних системах (MS Windows, Linux, MacOS).

При розробці відповідно до календарного плану було виконано

проектування окремих модулів системи, найбільше часу було витрачено на розробку графічного інтерфейсу і налагодження роботи системи. Із усіх функцій побудова графіків «Тесту мембрани» є дещо новою у цьому напрямку. Наявні програмні комплекси виконують їх побудову, але в режимі реального часу, але не для записаних раніше файлів. Не менш зручною є можливість побачити якість виконаних реєстрацій і після завершення експерименту.

Реалізована система дає можливість без необхідності навчання наукового персоналу одразу виконувати необхідні завдання. Час, який було затрачено на розробку системи, компенсується подальшим зростанням ефективності роботи відділу. Зважаючи на це, можна зробити висновок про економічну доцільність застосування даного програмного продукту для застосування до даних експериментів методу patch-clamp.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ГОСТ 19.701-90 (ИСО 5807-85), Единая система программной документации, СХЕМЫ АЛГОРИТМОВ, ПРОГРАММ ДАННЫХ И СИСТЕМ, Условные обозначения и правила выполнения, Москва
2. Межгосударственный стандарт, ГОСТ 2.104—2006, Единая система конструкторской документации, Основные надписи
3. SIGNAL ANALYSIS TIME, FREQUENCY, SCALE, AND STRUCTURE, Ronald L. Allen, Duncan W. Mills, IEEE Press, 445 Hoes Lane, Piscataway, NJ 08854, ISBN: 0-471-23441-9
4. Підручник мови Python, Ґвідо ван Россум, Фред Л. Дарк Молодший, переклад Сергій Кузьменко, 2005
5. Axon Binary File Format User Guide, Molecular Devices, LLC, Sunnyvale, California, United States of America, February 2013
6. Axon Binary File Format User Guide Version 2.0.9, Sunnyvale, California, United States of America, MolecularDevices,LLC, November 2017
7. Desai NS, Gray R, Johnston D. A Dynamic Clamp on Every Rig, eNeuro, 2017 Oct 23; 4(5): ENEURO.0250-17.2017
8. Digidata 1320A/1321A Operator’s Manual, Axon Instruments, Inc, Part Number 2500-130, March 1999, USA
9. Д. А. Маслов, И. В. Меркурьев, Линеаризация колебаний резонатора волнового твердотельного гироскопа и сил электростатических датчиков управления, Нелинейная динам., 2017, 413–421
10. Astrid A. Prinz, Robert H. CudmoreDynamic clamp, 2011, Scholarpedia, 6(5):1470
11. Axon pCLAMP 11 Software Suite, Molecular Devices LLC, Printed in USA 11/17

12. HumSilencer: A smart and simple feature fore liminating line-frequency noises across a wide range of electrophysiological applications, Molecular Devices, LLC, 10/19 EU
13. MultiClamp 700B Microelectrode Amplifier, Molecular Devices, LLC, 11/18 0120-1421E, USA
14. The Axon Guide : Electrophysiology and Biophysics Laboratory Techniques - Third Edition, 1-2500-0102 D, Molecular Devices, LLC, Sunnyvale, California, United States of America, February 2012
15. THE PATCH-CLAMP TECHNIQUE EXPLAINED AND EXERCISED WITH THE USE OF SIMPLE ELECTRICAL EQUIVALENT CIRCUITS, Dirk L. Ypey & Louis J. DeFelice, Department of Neurophysiology Leiden University Medical Center (LUMC), Leiden, The Netherlands, Department of Pharmacology, Vanderbilt University Medical Center Nashville, USA
16. Understanding the cell as an electrical circuit, Lea Goetz, Christian Wilms, UCL, London, UK, Scientifica Ltd, Uckfield, UK
17. Модель Ходжкіна — Хакслі: сайт, URL - https://uk.wikipedia.org/wiki/Модель_Ходжкіна_—_Хакслі
18. Пропрієтарне програмне забезпечення: сайт, URL - https://uk.wikipedia.org/wiki/Пропрієтарне_програмне_забезпечення
19. Середнє зважене - Вікіпедія: сайт, URL - https://uk.wikipedia.org/wiki/Середнє_зважене
20. Исключения в python. Конструкция try - except для обработки исключений - Python 3 для начинающих: сайт, URL - https://pythonworld.ru/tipy-dannyx-v-python/isklyucheniya-v-python-konstrukciya-try-except-dlya-obrabotki-isklyuchenij.html
21. Мембранный потенциал: сайт, URL - https://ru.wikipedia.org/wiki/Мембранный_потенциал
22. Модель ФитцХью — Нагумо: сайт, URL - https://ru.wikipedia.org/wiki/Модель_ФитцХью_—_Нагумо

23. Нормальное распределение - Википедия: сайт, URL - https://ru.wikipedia.org/wiki/Нормальное_распределение
24. нормальное распределение - SMART-LAB: сайт, URL - https://smart-lab.ru/finansoviy-slovar/нормальное%20распределение
25. Оформление списка литературы по ГОСТу - Kursach37: сайт, URL - http://kursach37.com/oformlenie-spiska-literatury-po-gost
26. Почему Python?: сайт, URL - https://khashtamov.com/ru/why-python/
27. Acquisition Hardware and Software PClamp Package from Molecular Devices-Digidata 1440+ pClamp 10, Warner Instruments: сайт, URL - http://www.harvardapparatus.com/media/harvar/pdf/W3_5_170.pdf
28. Axon pCLAMP 11 Electrophysiology Data Acquisition & Analysis Software, Published 11/28/2017, Updated 10/11/2019: сайт, URL - https://mdc.custhelp.com/app/answers/detail/a_id/20260/%E2%88%BC/axon%E2%84%A2-pclamp%E2%84%A2-11-electrophysiology-data-acquisition-%26-analysis-software
29. Configuration Scope - Magento Open Source, 1.9.x: сайт, URL - https://docs.magento.com/m1/ce/user_guide/configuration/scope.html
30. Deflate: сайт, URL - https://ru.wikipedia.org/wiki/Deflate
31. Delta encoding: сайт, URL - https://en.wikipedia.org/wiki/Delta_encoding
32. Electrophysiology - Molecular Devices: сайт, URL - https://www.moleculardevices.com/technology/electrophysiology
33. Encapsulated PostScript: сайт, URL - https://uk.wikipedia.org/wiki/Encapsulated_PostScript
34. Errors and Exceptions - The Python Tutorial: сайт, URL - https://docs.python.org/3/tutorial/errors
35. Gaussian filter - Wikipedia: сайт, URL - https://en.wikipedia.org/wiki/Gaussian_filter
36. GIF: сайт, URL - https://uk.wikipedia.org/wiki/GIF

37. HumSilencer Technology - Molecular Devices: сайт, URL - https://www.moleculardevices.com/technology/humsilencer
38. Markov model: сайт, URL - https://en.wikipedia.org/wiki/Markov_model
39. Membrane Test Theory and Analaysis Techniques: сайт, URL - https://github.com/swharden/pyABF/blob/master/docs/advanced/v1%20cookbook/memtest-simulation.ipynb
40. Origin (программа) : сайт, URL - https://ru.wikipedia.org/wiki/Origin_(программа)
41. Origin and OriginPro - OriginLab: сайт, URL - ttps://www.originlab.com/index.aspx?go=PRODUCTS/Origin
42. Patch clamp - Wikipedia: сайт, URL - https://en.wikipedia.org/wiki/Patch_clamp
43. pCLAMP 11 Software Suite: сайт, URL - https://www.moleculardevices.com/products/axon-patch-clamp-system/acquisition-and-analysis-software/pclamp-software-suite
44. PNG: сайт, URL - https://uk.wikipedia.org/wiki/PNG
45. PostScript: сайт, URL - https://uk.wikipedia.org/wiki/PostScript
46. Python: сайт, URL - https://uk.wikipedia.org//wiki/Python
47. Python Try Except - THE WORLD'S LARGESTWEB DEVELOPER SITE: сайт, URL - https://www.w3schools.com/python/python_try_except.asp
48. Questions and Answers - Synaptosoft, Inc.: сайт, URL - http://www.synaptosoft.com/MiniAnalysis/Features.html
49. Statistics - OriginLab: сайт, URL - https://www.originlab.com/index.aspx?go=Products/Origin/Statistics
50. SVG: сайт, URL - https://ru.wikipedia.org/wiki/SVG
51. Voltage clamp: сайт, URL - https://en.wikipedia.org/wiki/Voltage_clamp
52. What is the current-clamp method? : сайт, URL - https://www.moleculardevices.com/applications/patch-clamp-lectrophysiology/what-current-clamp-method

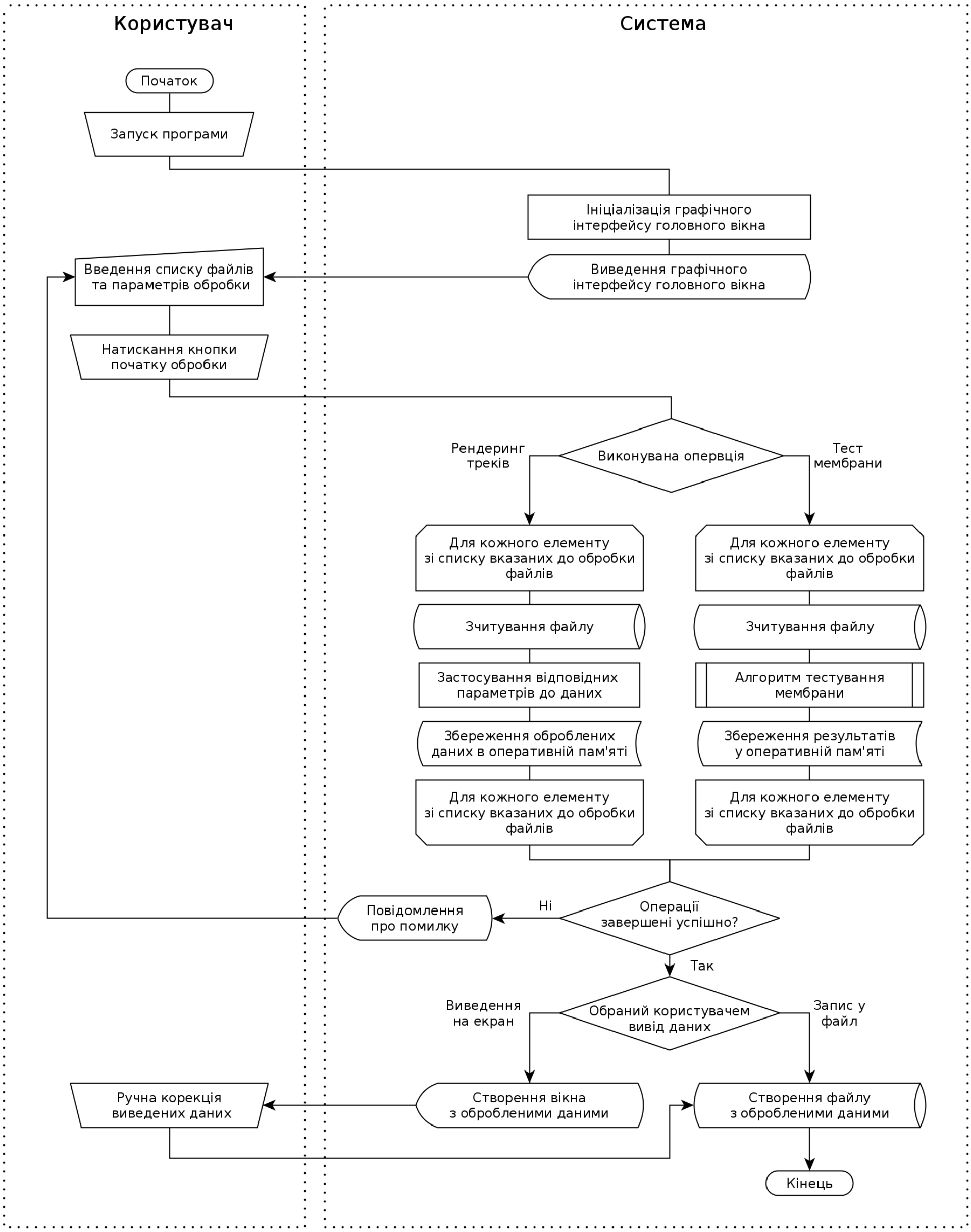
ДОДАТОК 1

Система обробки та візуалізації електрофізіологічних даних

Схеми алгоритмів структурні

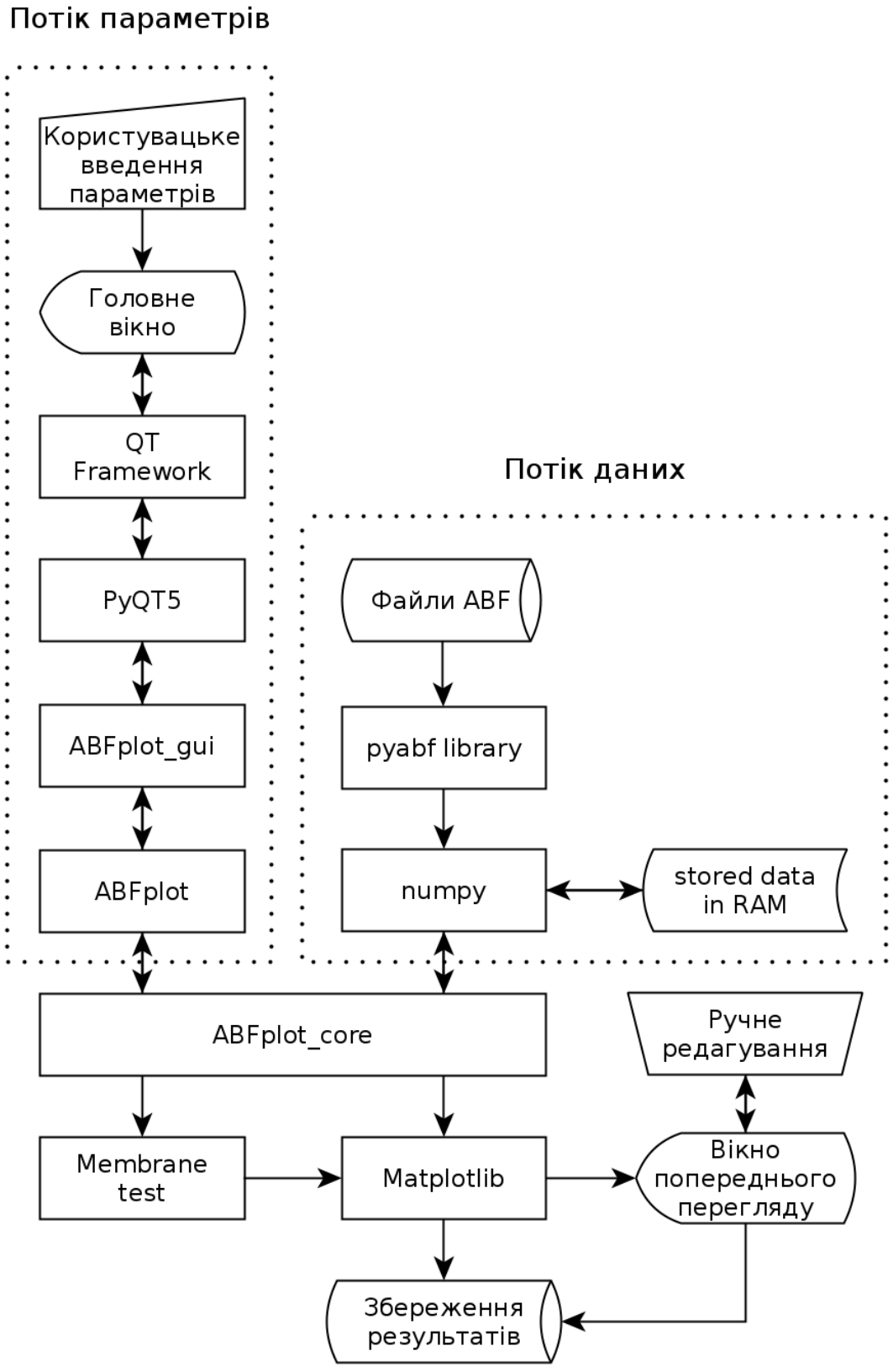
Аркушів 4

Київ - 2020

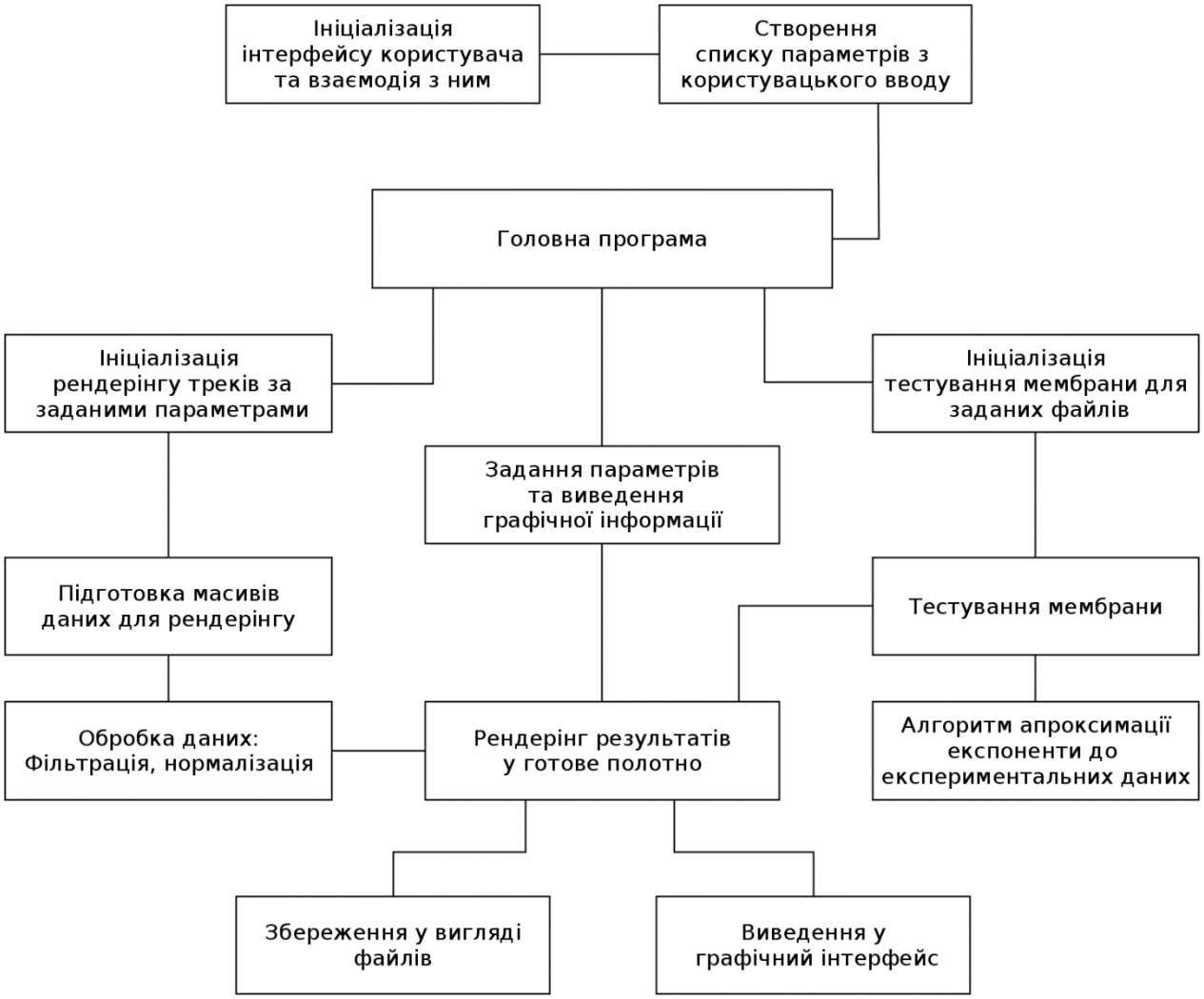


Підпис і дата | Інв. № дубл. | Взам. інв. № | Підпис і дата | Інв. № оригі.

Зм.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.467800.004 Д1		
Розробив		Сабецька			Система обробки та візуалізації електрофізіологічних даних Схема роботи системи	Літ.	Маса / Масштаб
Перевірив		Сімоненко				Аркуш	Аркушів 1
Н. контр.		Сімоненко				НТУУ «КПІ ім. Ігоря Сікорського», ФІОТ ІП-62	
Затвердив		Стіренко					



					ІАЛЦ.467800.004 Д2	
Зм.	Арк.	№ докум.	Підпис	Дата	Система обробки та візуалізації електрофізіологічних даних Схема взаємодії модулів	Літ. / Маса / Масштаб
Розробив		Сабецька				
Перевірив		Сімоненко				Аркуш / Аркушів 1
Н. контр.		Сімоненко				НТУУ «КПІ ім. Ігоря Сікорського», ФІОТ ІП-62
Затв.		Стіренко				



Зм.	Арк.	№ докум.	Підпис	Дата	ІАЛЦ.467800.004 ДЗ			
Разробив		Сабецька			Система обробки та візуалізації електрофізіологічних даних	Літ.	Маса	Масштаб
Перевірив		Сімоненко						
					Структурна схема програми	Аркуш	Аркушів 1	
						НТУУ «КПІ ім. Ігоря Сікорського», ФІОТ ІП-62		
Н. контр.		Сімоненко						
Затв.		Стіренко						

ДОДАТОК 2

Система обробки та візуалізації електрофізіологічних даних

Лістинг програми

Аркушів 10

Київ - 2020

```
#!/usr/bin/env python3

import sys
import numpy as np
import matplotlib.pyplot as plt
import pyabf
import pyabf.filter
import pyabf.tools.memtest
from itertools import zip_longest

# Задання глобального масиву вертикальних зміщень
OFFSETS = [0.0, 0.0, 0.0, 0.0]

# Функція виведеня окремо взятого треку
def makeline(ABF_FILE, SINCE, STIM, EXCLUDE_SWEEPS, BASELINE, SIGMA,
             OFFSET_Y, LINE_WIDTH, ALPHA, MIN_X, MAX_X, channel=0,
             plotonly=''):

    global OFFSETS

    # Перехоплення помилки відсутнього файлу
    try:
        # Відкривання abf файлу
        abf = pyabf.ABF(ABF_FILE + '.abf')
    except ValueError:
        raise FileNotFoundError

    # Визначення загальної кількості треків
    sweep_count = abf.sweepCount
    pyabf.filter.gaussian(abf, 0)
```

```
 # Задання інтенсивності фільтрування даних
    pyabf.filter.gaussian(abf, SIGMA)

    if SINCE == 0:
        SINCE = sweep_count + 1

    # Створення груп треків, всі, що за номером після значення SINCE -
    # йдуть у групу B
    sweep_a = list(range(0, SINCE-1))
    sweep_b = list(range(SINCE-1, sweep_count))

    # Об'явлення масивів значень x та y для групи треків
    sweep_a_x_list = np.array([], float)
    sweep_a_y_list = np.array([], float)
    sweep_b_x_list = np.array([], float)
    sweep_b_y_list = np.array([], float)

    # Створення масивів для рендерингу треків
    for sweep_n_a, sweep_n_b in zip_longest(sweep_a, sweep_b,
fillvalue=None):

        # Створення масиву з треків виключно групи B
        if plotonly != 2 and sweep_n_a != None:

            # Обрання конкретного треку з файлу
            abf.setSweep(sweep_n_a, channel, baseline=BASELINE)

            # Поміщення треку в масив тільки якщо його не виключено
            if sweep_n_a + 1 not in EXCLUDE_SWEEPS:

                # Якщо список з даними треків пустий - додати першим
                # елементом ділянку треку
```

```
            if list(sweep_a_x_list) == []:
                sweep_a_x_list = [abf.sweepX[MIN_X:MAX_X]]
                sweep_a_y_list = [abf.sweepY[MIN_X:MAX_X]]
            # Інакше - додати наступним елементом ділянку треку
            else:
                sweep_a_x_list = np.vstack(
                    (sweep_a_x_list, abf.sweepX[MIN_X:MAX_X]))
                sweep_a_y_list = np.vstack(
                    (sweep_a_y_list, abf.sweepY[MIN_X:MAX_X]))

    # Створення масиву з треків виключно групи A
    if plotonly != 1 and sweep_n_b != None:

        # Обрання конкретного треку з файлу
        abf.setSweep(sweep_n_b, channel, baseline=BASELINE)

        # Перенесення треку в масив тільки якщо його не виключено
        if sweep_n_b + 1 not in EXCLUDE_SWEEPS:
            # Якщо список з даними треків пустий - додати першим
            # елементом ділянку треку
            if list(sweep_b_x_list) == []:
                sweep_b_x_list = [abf.sweepX[MIN_X:MAX_X]]
                sweep_b_y_list = [abf.sweepY[MIN_X:MAX_X]]
            # Інакше - додати наступним елементом ділянку треку
            else:
                sweep_b_x_list = np.vstack(
                    (sweep_b_x_list, abf.sweepX[MIN_X:MAX_X]))
                sweep_b_y_list = np.vstack(
                    (sweep_b_y_list, abf.sweepY[MIN_X:MAX_X]))

# Додавання до кожного значення Y вертикальне зміщення OFFSET_Y
def makeoffset(sweep_y_list):
```

```
        for i in range(0, len(sweep_y_list)):
            OFFSETS[1] = i * OFFSET_Y
            sweep_y_list[i] += OFFSETS[1]

    makeoffset(sweep_a_y_list)
    makeoffset(sweep_b_y_list)

    # Створення кінцевого списку для рендерингу
    data_to_plot = zip_longest(
        sweep_a_x_list, sweep_a_y_list, sweep_b_x_list, sweep_b_y_list,
fillvalue=[])

    # Рендеринг треків по черзі по одному з кожної групи
    for sweep_a_x_data, sweep_a_y_data, sweep_b_x_data, sweep_b_y_data in
data_to_plot:

        if plotonly != 2:
            plt.plot(sweep_a_x_data, sweep_a_y_data, lw=LINE_WIDTH,
                     alpha=ALPHA, color='black' if STIM == 1 else 'red')
        if plotonly != 1:
            plt.plot(sweep_b_x_data, sweep_b_y_data, lw=LINE_WIDTH,
                     alpha=ALPHA, color='black' if STIM == -1 else 'red')


# Створення підписів до осі абсцис
def labels_x(MIN_X, MAX_X, FREQ):

    labels = []
    for i in range(MIN_X, MAX_X + 500, 100):
        if (i*2 / FREQ * 1000) % 25 == 0:
            labels.append(int(i*2 / FREQ * 1000))
        else:
```

```
            labels.append('')
    return labels

# Масштабування по осям абсцис та ординат
def makefigure(rows_n, cols_n, plot_n, FREQ, MIN_X, MAX_X, MIN_Y=0,
MAX_Y=0, yaxis=True):
    axis = [MIN_X/FREQ, MAX_X/FREQ] + [MIN_Y, MAX_Y]
    plt.subplot(rows_n, cols_n, plot_n)

    if not yaxis:
        plt.yticks([])
    if yaxis:
        plt.axis(axis)

# Функція, що компонує готові рисунки
def plot(ABF_FILE, SINCE, STIM, EXCLUDE_SWEEPS, CHANNEL, BASELINE, SIGMA,
DESCR, OFFSET_Y, TWO_WIN, LINE_WIDTH, ALPHA, FIGURE_W, FIGURE_H, DPI,
FREQ, SHOW, SAVE, SAVE_FORMAT, MIN_X, MAX_X, MIN_Y, MAX_Y):

    # Створення нового рисунку
    plt.figure(num=None, figsize=(FIGURE_W, FIGURE_H),
               dpi=DPI if SAVE else None)

    # Виведення для одного з каналів
    if CHANNEL == 0 or CHANNEL == 1:

        if TWO_WIN == True:

            for i in range(1, 3):

                makefigure(1, 2, i, FREQ, MIN_X, MAX_X, yaxis=False)
```

```
            for i in range(len(ABF_FILE)):
                makeline(ABF_FILE[i], SINCE[i], STIM[i],
                       EXCLUDE_SWEEPS[i], BASELINE, SIGMA, OFFSET_Y,
                       LINE_WIDTH, ALPHA, MIN_X, MAX_X,
                       channel=CHANNEL, plotonly=i)

    else:

        if OFFSET_Y == 0:
            makefigure(1, 1, 1, FREQ, MIN_X, MAX_X, MIN_Y, MAX_Y)
        else:
            makefigure(1, 1, 1, FREQ, MIN_X, MAX_X, yaxis=False)

        for i in range(len(ABF_FILE)):
            makeline(ABF_FILE[i], SINCE[i], STIM[i],
                  EXCLUDE_SWEEPS[i], BASELINE, SIGMA, OFFSET_Y,
                  LINE_WIDTH, ALPHA, MIN_X, MAX_X, channel=CHANNEL)

# Виведення для двох каналів одночасно
if CHANNEL == 2:

    if OFFSET_Y == 0:

        for i in range(2):

            makefigure(2, 1, i+1, FREQ, MIN_X, MAX_X, MIN_Y, MAX_Y)

            for i in range(len(ABF_FILE)):
                makeline(ABF_FILE[i], SINCE[i], STIM[i],
                       EXCLUDE_SWEEPS[i], BASELINE, SIGMA, OFFSET_Y,
                       LINE_WIDTH, ALPHA, MIN_X, MAX_X, channel=i)
```

```
        else:

            for i in range(2):

                makefigure(2, 1, i+1, FREQ, MIN_X, MAX_X, yaxis=False)

                for i in range(len(ABF_FILE)):
                    makeline(ABF_FILE[i], SINCE[i], STIM[i],
                           EXCLUDE_SWEEPS[i], BASELINE, SIGMA, OFFSET_Y,
                           LINE_WIDTH, ALPHA, MIN_X, MAX_X, channel=i)

    # Створення  рисунку
    plt.suptitle('\n' + ABF_FILE[0] + ' ' + DESCR, size=10)
    plt.tight_layout()

    # Збереження рисунку у форматі, вказаному в SAVE_FORMAT
    if SAVE:
        plt.savefig(ABF_FILE[0] + '.' + SAVE_FORMAT)
    if SHOW:
        plt.show()


def membrane_test(ABF_FILE, FIGURE_W, FIGURE_H):

    # Перехоплення помилки відсутнього файлу
    try:
        # Відкривання abf файлу
        abf = pyabf.ABF(ABF_FILE + '.abf')
    except ValueError:
        raise FileNotFoundError

    memtest = pyabf.tools.memtest.Memtest(abf)
```

```
# Створення нового рисунку
fig = plt.figure(figsize=(FIGURE_W, FIGURE_H))

# Виведення значень струму утримання (Ih)
ax1 = fig.add_subplot(221)
ax1.grid(alpha=.2)
ax1.plot(abf.sweepTimesMin, memtest.Ih.values,
         ".", color='C0', alpha=.7, mew=0)
ax1.set_title(memtest.Ih.name)
ax1.set_ylabel(memtest.Ih.units)

# Виведення значень мембранного опору(Rm)
ax2 = fig.add_subplot(222)
ax2.grid(alpha=.2)
ax2.plot(abf.sweepTimesMin, memtest.Rm.values,
         ".", color='C3', alpha=.7, mew=0)
ax2.set_title(memtest.Rm.name)
ax2.set_ylabel(memtest.Rm.units)

# Виведення значень опору доступу (Ra)
ax3 = fig.add_subplot(223)
ax3.grid(alpha=.2)
ax3.plot(abf.sweepTimesMin, memtest.Ra.values,
         ".", color='C1', alpha=.7, mew=0)
ax3.set_title(memtest.Ra.name)
ax3.set_ylabel(memtest.Ra.units)

# Виведення значень мембранної ємності (Cm)
ax4 = fig.add_subplot(224)
ax4.grid(alpha=.2)
```

```
ax4.plot(abf.sweepTimesMin, memtest.CmStep.values,
         ".", color='C2', alpha=.7, mew=0)
ax4.set_title(memtest.CmStep.name)
ax4.set_ylabel(memtest.CmStep.units)

# Вивести значення на осі абсцис
for ax in [ax1, ax2, ax3, ax4]:
    ax.margins(0, .9)
    ax.set_xlabel("Experiment Time (minutes)")
    for tagTime in abf.tagTimesMin:
        ax.axvline(tagTime, color='k', ls='--')

# Вивести рисунок
plt.tight_layout()
plt.suptitle(ABF_FILE)
plt.show()
```

ДОДАТОК 3

Система обробки та візуалізації електрофізіологічних даних

Опис програми

Аркушів 6

Київ - 2020

Анотація

У програмному комплексі ABFplot реалізована пакетна обробка та візуалізвція електрофізіологічних даних. Програма написана на мові Python третьої версії. Графічний інтерфейс користувача реалізований за допомогою QT Frameworks. Ці фактори роблять її незалежною від платформи та здатною працювати на більшості сучасних операційних системах. Програма виконує ряд специфічних задач та підходить для використання в науково-дослідних установах студентами, аспірантами та науковцями. Відкритий вихідний код дозволяє кожному вносити свої зміни в її код для модифікації під власні потреби.

Зміст

1. Загальні відомості

Система обробки та візуалізації електрофізіологічних даних має назву ABFplot. Написана на мові Python версії 3. Програма кросплатформна, а також не залежить від операційної системи. Вимогою для роботи програми є можливість встановлення на робочий комп'ютер Python не нижче третьої версії і можливості встановлення фреймворку QT 5, необхідного для роботи графічного інтерфейсу користувача. Система використовує:

• pyABF - пакет Python, який забезпечує зчитування електро-фізіологічних даних з файлів Axon Binary Format (ABF) версій 1 і 2;

• NumPy – бібліотека, що забезпечує підтримку багатовимірних масивів і містить безліч функцій і операторів для роботи з ними;

• Matplotlib – бібліотека для візуалізації даних двовимірною і трьохвимірною графікою;

• PyQt5 – модуль для взаємодії графічного фреймворку Qt з мовою Pyton;

• sys – забезпечує функції і константи для взаємодії з інтерпретатором Pyton;

• itertools – модуль для створення циклів ітерації.

Програма розрахована на використання науковцями які працюють методикою patch-clamp з обладнанням від Molecular Devices.

2. Функціональне призначення

. Головне функціональне призначення програми — пакетна обробка електрофізіолгічних даних, що знаходяться у файлах .abf, та рендеринг графічної інформації. Для коректного зчитування параметрів в програмі виконується перехоплення виключень. Так само відбувається виявлення пошкоджених файлів .abf. Основні функції винесені в набір налаштувань за замовчуванням, вони відповідають найбільш розповсюдженим потребам користувача.

Програма дозволяє робити попередній перегляд та ручну корекцію візуалізованих графічних даних.

3. Опис логічної структури

Програма складається з трьох модулів: abfplot, abfplot_core та abfplot_gui.

Також використовуються такі сторонні модулі, як pyabf, numpy, Matplotlib, PyQt5 (див. Додаток 1. Схема взаємодії модулів).

Модуль ABFplot є головним. Він відповідає за ініціалізацію програми та передачу параметрів від графічного інтерфейсу до функцій обробки даних, необхідних для подальшого рендерингу графічної інформації.

Модуль ABFplot_core містить всі функції з обробки даних, та виконує рендеринг графічної інформації. Для роботи з файлами .abf використовується відкрита бібліотека pyabf. За допомогою неї вхідні дані з бінарних файлів перетворюються у масиви даних, з якими може працювати програма. Робота з масивами даних відбувається через модуль numpy, який значно пришвидшує роботу, оскільки являється спеціально оптимізованим для швидкої обробки великих масивів даних. Рендеринг кінцевих графічних зображень виконується через бібліотеку Matplotlib.

Модуль ABFplot_gui містить параметри графічного інтерфейсу та відповідає за його ініціалізацію через фреймворк QT 5. Робить він це через сторонній модуль PyQt5, який являє собою оболонку для фреймворку QT та відповідає за його інтеграцію з Python.

4. Виклик та завантаження

Програма розповсюджується як у вигляді архіву з вихідним кодом, так і в скомпільованому стані одним виконуваним бінарним файлом. Для

операційної системи Windows формат виконуваного файлу — EXE, для Linux та інших UNIX-подібних систем формат виконуваного файлу - ELF.

Щоб викликати програму зі скомпільованого бінарного файлу достатньо запустити її подвійним кліком курсору на піктограмі файлу ABFplot. При запуску програми у вигляді бінарного файлу, всі необхідні бібліотеки вже є вбудованими у виконуваний файл.

Коли програму викликають з вихідного коду, необхідно запустити файл abfplot.pyw через графічний інтерфейс операційної системи або через командний термінал. Слід зауважити, що у випадку виклику програми з вихідного коду слід мати встановленими в системі середу виконування Python версії 3, а також мати встановленими такі необхідні модулі Python: pyabf, numpy, Matplotlib, PyQt5

5. Вхідні дані

Вхідні дані представляють собою бінарні файли формату Axon Binary Format (ABF) версій 1 і 2, що створюються програмним забезпеченням pCLAMP, яке працює напряму з електрофізіологічним обладнанням.

6. Вихідні дані

Вихідні дані представляють собою файли векторних (SVG, EPS) та растрових (PNG) графічних форматів. Також можливе створення електронних документів PDF.